

1/5/2

DIALOG(R) File 351:Derwent WPI

(c) 2000 Derwent Info Ltd. All rts. reserv.

010144072 \*\*Image available\*\*

WPI Acc No: 1995-045323/199507

XRPX Acc No: N95-035701

**Distributed computation based on movement, execution and insertion of processes in network - moving agent process between places and transporting only class definition which are not already at second location**

Patent Assignee: GEN MAGIC INC (GEMA-N)

Inventor: HELGESON C S; STEEDMAN D A; WHITE J E

Number of Countries: 049 Number of Patents: 008

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
EP 634719	A2	19950118	EP 94305058	A	19940708	199507 B
WO 9502219	A1	19950119	WO 94US7397	A	19940708	199509
AU 9472164	A	19950206	AU 9472164	A	19940708	199518
JP 7182174	A	19950721	JP 94179767	A	19940708	199538
JP 7509799	W	19951026	WO 94US7397	A	19940708	199551
			JP 95504073	A	19940708	
EP 634719	A3	19960103	EP 94305058	A	19940708	199620
US 5603031	A	19970211	US 9390521	A	19930708	199712
US 6016393	A	20000118	US 9390521	A	19930708	200011
			US 97798675	A	19970210	

Priority Applications (No Type Date): US 9390521 A 19930708; US 97798675 A 19970210

Cited Patents: Jnl.Ref; EP 495310; EP 546809; US 4575797; US 5129083; WO 9110191; US 5079695; US 5093914; US 5261080; US 5303375; US 5307490; US 5327559; US 5339430

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
-----------	------	--------	----------	--------------

EP 634719	A2	E 640	G06F-009/46	
-----------	----	-------	-------------	--

Designated States (Regional): DE ES FR GB IT NL SE

US 6016393	A	G06F-013/00	Cont of application US 9390521
			Cont of patent US 5603031

WO 9502219	A1	E 797	G06F-013/00	
------------	----	-------	-------------	--

Designated States (National): AT AU BB BG BR BY CA CH CN CZ DE DK ES FI GB GE HU JP KG KP KR KZ LK LU LV MD MG MN MW NL NO NZ PL PT RO RU SD SE SI SK TJ TT UA UZ VN

Designated States (Regional): OA

AU 9472164	A	G06F-013/00	Based on patent WO 9502219
------------	---	-------------	----------------------------

JP 7182174	A	309 G06F-009/44	
------------	---	-----------------	--

JP 7509799	W	261 G06F-013/00	Based on patent WO 9502219
------------	---	-----------------	----------------------------

US 5603031	A	293 G06F-013/00	
------------	---	-----------------	--

EP 634719	A3	G06F-009/46	
-----------	----	-------------	--

Abstract (Basic): EP 634719 A

The method for implementing remote programming in a computer network involves defining several object oriented classes including agent and plane classes. Instructions for a computer process are formed which include object oriented classes, subclasses and a go operation.

The instructions are interpreted on a processor in the computer network. In response to the go operation, an agent operation is transported to a place process, where the agent process is a member of the agent class and the place process is a member of the place class. Preferably, the go operation is formed within the agent process.

USE/ADVANTAGE - Object oriented program. Produces autonomous clones. Increases efficiency.

Dwg.4/71

Title Terms: DISTRIBUTE; COMPUTATION; BASED; MOVEMENT; EXECUTE; INSERT; PROCESS; NETWORK; MOVE; AGENT; PROCESS; PLACE; TRANSPORT; CLASS; DEFINE; SECOND; LOCATE

Derwent Class: T01

International Patent Class (Main): G06F-009/44; G06F-009/46; G06F-013/00

1c925 U.S. PRO

09/695195

10/25/00

A3  
#3

International Patent Class (Additional): G06F-009/00; G06F-015/16  
File Segment: EPI



(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平7-182174

(43)公開日 平成7年(1995)7月21日

(51)Int.Cl. <sup>8</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 9/44	5 3 5	9193-5B		
13/00	3 5 5	7368-5B		
15/16	3 7 0 N			

審査請求 未請求 請求項の数165 F D (全 309 頁)

(21)出願番号 特願平6-179767

(22)出願日 平成6年(1994)7月8日

(31)優先権主張番号 08/090,521

(32)優先日 1993年7月8日

(33)優先権主張国 米国 (US)

(71)出願人 594129286

ジェネラル マジック, インク.  
General Magic, Inc.  
アメリカ合衆国 カリフォルニア州  
94086, サニーベイル, ノース マリー  
アベニュー 420

(72)発明者 ジェイムス イー. ホワイト

アメリカ合衆国 カリフォルニア州  
94070, サン カルロス, ワイクム  
アベニュー 112

(74)代理人 弁理士 小池 晃 (外2名)

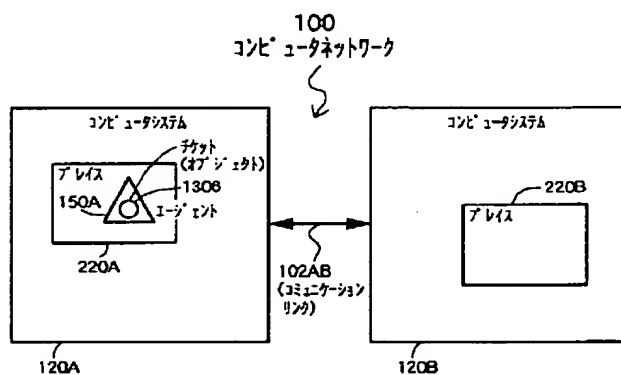
最終頁に続く

(54)【発明の名称】 リモートプログラミングの実施方法

(57)【要約】

【構成】 コンピュータネットワークにおけるリモートプログラミングの実施方法は、エージェントクラス及びプレイスクラスを含む複数のオブジェクト指向クラスを定義し、オブジェクト指向クラス、オブジェクト指向クラスのサブクラス及び go オペレーションを含む、コンピュータプロセスのためのインストラクションを形成し、コンピュータネットワーク中のプロセッサによって前記インストラクションをインタープリートし、go オペレーションに応答して、エージェントプロセスがプレイスプロセスに転送され、エージェントプロセスが、エージェントクラスの1つのメンバであり、プレイスプロセスはプレイスクラスの1つのメンバである。

【効果】 本発明のインストラクションは、オペレーティングシステム及びハードウェアが異なるコンピュータシステムにおいても、移動され実行できる。



## 【特許請求の範囲】

【請求項1】 エージェントクラス及びプレイスクラスを含む複数のオブジェクト指向クラスを定義し、前記オブジェクト指向クラス、該オブジェクト指向クラスのサブクラス及び go オペレーションを含む、コンピュータプロセスのためのインストラクションを形成し、該コンピュータネットワーク中のプロセッサによって前記インストラクションをインタープリートし、

前記 go オペレーションにตอบสนองして、エージェントプロセスがプレイスプロセスに転送され、前記エージェントプロセスが、前記エージェントクラスの1つのメンバであり、前記プレイスプロセスは前記プレイスクラスの1つのメンバであることを特徴とするコンピュータネットワークにおけるリモートプログラミングの実施方法。

【請求項2】 前記エージェントプロセスにおいて前記 go オペレーションを形成することを特徴とする請求項第1項記載のリモートプログラミングの実施方法。

【請求項3】 前記プレイスのサブプレイスを形成し、前記サブプレイスは前記プレイスクラスの1つのメンバであり、前記プレイスは前記サブプレイスの1つのスーパープレイスであることを特徴とする請求項第1項記載のリモートプログラミングの実施方法。

【請求項4】 前記エージェントプロセスをオブジェクトのオーナーとして指定することを特徴とする請求項第1項記載のリモートプログラミングの実施方法。

【請求項5】 前記オブジェクト中にダイジェストを形成することを特徴とする請求項第4項記載のリモートプログラミングの実施方法。

【請求項6】 前記オブジェクトの代わりに前記プレイスプロセスにおいて第2のオブジェクトを相互変換し、前記第2のオブジェクトは、前記ダイジェストと同等の第2のダイジェストを有し、前記第1のオブジェクトは前記プレイスプロセスに転送されないことを特徴とする請求項第5項記載のリモートプログラミングの実施方法。

【請求項7】 前記プレイスプロセスをチケット手段によって指定することを特徴とすることを請求項第1項記載のリモートプログラミングの実施方法。

【請求項8】 前記チケット手段を前記エージェントプロセス内において形成することを特徴とする請求項第7項記載のリモートプログラミングの実施方法。

【請求項9】 前記チケット手段中において前記プレイスプロセスのアドレスを形成することを特徴とする請求項第7項記載のリモートプログラミングの実施方法。

【請求項10】 前記アドレスがテレアドレスであることを特徴とする請求項第9項記載のリモートプログラミングの実施方法。

【請求項11】 前記チケット手段中において前記プレイスプロセスのネームを形成することを特徴とする請求項第7項記載のリモートプログラミングの実施方法。

【請求項12】 前記ネームがテレネームであることを特徴とする請求項第11項記載のリモートプログラミングの実施方法。

【請求項13】 クラスオブジェクトのクラスを規定し、

クラスオブジェクトを形成し、

前記クラスオブジェクトは、(i) クラスオブジェクトの前記クラスの1つのメンバであり、(i i) 前記オブジェクト指向クラスの選択された1つのもののサブクラスである第1のクラスを規定し、

前記第1のクラスの1つのメンバであるオブジェクトを形成し、

前記エージェントプロセスは、前記オブジェクトを所有し、前記エージェントプロセスを前記プレイスプロセスに転送することが、前記オブジェクト及び前記クラスオブジェクトを前記プレイスプロセスに転送することを含むことを特徴とする請求項第1項記載のリモートプログラミングの実施方法。

【請求項14】 エージェントクラスとプレイスクラスとを含む複数のオブジェクト指向クラスを定義し、前記オブジェクト指向クラス、前記オブジェクト指向クラスのサブクラス及び send オペレーションを含むコンピュータプロセスのためのインストラクションを形成し、前記コンピュータのネットワーク中のプロセッサにおいて前記インストラクションをインタープリートする際に、

前記 send オペレーションにตอบสนองしてエージェントプロセスの1つのクローンが1つのプレイスプロセスに転送され、前記クローン及び前記エージェントプロセスは、各々前記エージェントクラスの1つのメンバであり、前記プレイスプロセスは前記プレイスクラスの1つのメンバであることを特徴とするコンピュータネットワークにおけるリモートプログラミングの実施方法。

【請求項15】 前記 send オペレーションを前記エージェントプロセス中において形成することを特徴とする請求項第14項記載のリモートプログラミングの実施方法。

【請求項16】 前記プレイスのサブプレイスを形成し、

前記サブプレイスは前記プレイスクラスの1つのメンバであり、前記プレイスは前記サブプレイスの1つのスーパープレイスであることを特徴とする請求項第14項記載のリモートプログラミングの実施方法。

【請求項17】 前記エージェントプロセスをオブジェクトのオーナーとして指定することを特徴とする請求項第14項記載のリモートプログラミングの実施方法。

【請求項18】 前記クロンの転送が前記プロセスのコピーを前記プレイスプロセスに転送することを特徴とする請求項第17項記載のリモートプログラミングの実施方法。

【請求項 19】 前記オブジェクト中にダイジェストを形成することを特徴とする請求項第 18 項記載のリモートプログラミングの実施方法。

【請求項 20】 前記オブジェクトと異なる第 2 のオブジェクトを前記ブレイスプロセスにおいて前記コピーと相互変換し、前記ブレイスプロセスにおいて前記第 2 のオブジェクトは、前記ダイジェストと同等の第 2 のダイジェストを持ち、前記コピーは前記ブレイスプロセスに転送されないことを特徴とする請求項第 19 項記載のリモートプログラミングの実施方法。

【請求項 21】 前記ブレイスプロセスをチケット手段によって指定することを特徴とする請求項第 14 項記載のリモートプログラミングの実施方法。

【請求項 22】 前記エージェントプロセス中において前記チケット手段を形成することを特徴とする請求項第 21 項記載のリモートプログラミングの実施方法。

【請求項 23】 前記ブレイスプロセスのアドレスを前記チケット手段中において形成することを特徴とする請求項第 21 項記載のリモートプログラミングの実施方法。

【請求項 24】 前記アドレスがテレアドレスであることを特徴とする請求項第 23 項記載のリモートプログラミングの実施方法。

【請求項 25】 前記チケット手段中において前記ブレイスプロセスの 1 つのネームを形成することを特徴とする請求項第 21 項記載のリモートプログラミングの実施方法。

【請求項 26】 前記ネームがテレネームであることを特徴とする請求項第 25 項記載のリモートプログラミングの実施方法。

【請求項 27】 本発明は、前記 send オペレーションに回答して前記エージェントプロセスの第 2 のクローンを第 2 のブレイスプロセスに転送し、前記第 2 のクローンは、前記クローンとは異なり、前記エージェントクラスの 1 つのメンバであり、前記第 2 のブレイスプロセスは前記ブレイスプロセスの 1 つのメンバであることを特徴とする請求項第 14 項記載のリモートプログラミングの実施方法。

【請求項 28】 前記第 1 のクローンを前記第 1 のブレイスプロセスに転送し、前記第 2 のクローンを前記第 2 のブレイスプロセスに転送するためには、前記コンピュータネットワークの複数のコンピュータの内の単一のコンピュータに前記第 1 及び第 2 のクローンを転送することを必要であることを定め、前記第 1 のクローンを前記単一のコンピュータに転送し、

前記単一のコンピュータにおいて前記第 1 のクローンから前記第 2 のクローンを形成することを特徴とする請求項第 27 項記載のリモートプログラミングの実施方法。

【請求項 29】 前記単一のコンピュータから前記ブレイスプロセスに前記第 2 のクローンを転送し、

前記単一のコンピュータから前記第 2 のブレイスプロセスに前記第 2 のクローンを転送することを特徴とする請求項第 28 項記載のリモートプログラミングの実施方法。

【請求項 30】 エージェントクラスを含む複数のオブジェクト指向クラスを定義し、前記オブジェクト指向クラス、前記オブジェクト指向クラスのサブクラス及び meet オペレーションを含むコンピュータプロセスのためのインストラクションを形成し、

前記コンピュータネットワーク中の 1 つのプロセッサにおいて前記インストラクションをインタープリートし、

ミーティングブレイスプロセスが前記 meet オペレーションに回答して、第 2 のエージェントプロセスへの第 1 のエージェントプロセスのアクセスを供与し、前記第 1 のエージェントプロセスへの前記第 2 のエージェントプロセスのアクセスも供与し、さらに、前記第 1 及び第 2 のエージェントプロセスが前記エージェントクラスのメンバであることを特徴とするコンピュータにおけるコンピュータのプロセス間通信方法。

【請求項 31】 前記ミーティングブレイスプロセスが前記複数のオブジェクト指向クラス中のブレイスクラスの 1 つのメンバであり、前記第 1 及び第 2 のエージェントプロセスが前記ミーティングブレイスプロセスを占有することを特徴とする請求項第 30 項記載のコンピュータのプロセス間通信方法。

【請求項 32】 第 2 のブレイスプロセスを形成し、第 2 のブレイスプロセスは前記ブレイスクラスの 1 つのメンバであり、前記第 2 のブレイスプロセスは前記ミーティングブレイスプロセスの 1 つのサブブレイスであり、前記ミーティングブレイスは前記第 2 のブレイスのスーパーブレイスであることを特徴とする請求項第 31 項記載のコンピュータのプロセス間通信方法。

【請求項 33】 エージェントクラス、ブレイスクラス及びチケットクラスを含む複数のオブジェクト指向クラスを定義し、各々が前記ブレイスクラスの 1 つのメンバである、前記コンピュータネットワーク中の複数のブレイスプロセスを形成し、

前記エージェントクラスの 1 つのメンバであって前記複数のブレイスプロセス中の第 1 のブレイスプロセスを占有する、エージェントプロセスを形成し、

前記チケットクラスの 1 つのメンバであり、前記複数のブレイスプロセス中の前記第 1 のブレイスプロセスから第 2 のブレイスプロセスへの前記エージェントプロセスの移動を含むトリップを定義するチケットを形成することを特徴とするコンピュータネットワークにおけるプロセスの運動制御方法。

【請求項 34】 ブレイスプロセスとパーミットクラス

とを含む複数のオブジェクト指向クラスを定義し、前記ブレイスクラスの1つのメンバであるプロセスを形成し、前記パーミットクラスの1つのメンバであって前記プロセスの1以上の能力を指定するパーミットを形成することを特徴とするコンピュータネットワーク中のプロセスの能力制御方法。

【請求項35】 前記プロセスクラス中に go オペレーションを定義し、

前記プロセスが前記 go オペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする請求項第34項記載のプロセスの能力制御方法。

【請求項36】 前記プロセスクラス中に send オペレーションを定義し、

前記プロセスが前記 send オペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする請求項第34項記載のプロセスの能力制御方法。

【請求項37】 前記プロセスクラス中において charge オペレーションを定義し、

前記プロセスが前記 charge オペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする請求項第34項記載のプロセスの能力制御方法。

【請求項38】 前記プロセスクラス中に terminate オペレーションを定義し、

前記プロセスが前記 terminate オペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする請求項第34項記載のプロセスの能力制御方法。

【請求項39】 前記プロセスが前記プロセスと異なった第2のプロセスを作り出すことができるか否かを前記パーミット中において指定し、

該第2のプロセスは前記プロセスクラスの1つのメンバであることを特徴とする請求項第34項記載のプロセスの能力制御方法。

【請求項40】 オブジェクト指向ブレイスクラスを定義し、

前記パーミット中に前記プロセスが前記ブレイスプロセスの複数のメンバを作り出すことができるか否かを前記パーミット中において指定することを特徴とする請求項第34項記載のプロセスの能力制御方法。

【請求項41】 前記プロセスが失敗した時に前記プロセスが再スタートされるか否かを前記パーミット中において指定することを特徴とする請求項第34項記載のプロセスの能力制御方法。

【請求項42】 前記プロセスに割り当てられた処理量を前記パーミット中において指定することを特徴とする請求項第34項記載のプロセスの能力制御方法。

【請求項43】 前記プロセスが終了する時間を前記パーミット中において指定することを特徴とする請求項第34項記載のプロセスの能力制御方法。

【請求項44】 前記プロセスクラス中において制限オペレーションを定義し、

前記クラス、前記クラスのサブクラス及び前記制限オペレーションを含むコンピュータプロセスのためのインストラクションを形成し、

前記コンピュータネットワーク中のプロセッサにおいて前記インストラクションをインタープリットし、

前記パーミットと異なる第2のパーミットが前記制限オペレーションに応答して形成され、

前記第2のパーミットは前記パーミットクラスの1つのメンバであり、前記プロセスの前記1以上の能力の1つのグループを指定し、

前記第2のパーミットによって指定された1以上の能力の前記グループに前記プロセスを制限することを特徴とする請求項第34項記載のプロセスの能力制御方法。

【請求項45】 コンピュータにおいて、

クラス(複数)の1つのクラスとサイテーション(複数)の1つのクラスを含む、複数のオブジェクト指向クラスを定義し、

前記複数のクラスの前記クラスのメンバである1以上のクラスオブジェクトを形成し、

前記第1のクラスオブジェクトを指定するとともに、前記クラスオブジェクトの内のどれが前記第1のオブジェクトに対して後方にコンパティブルであるかを指定するサイテーションを前記クラスオブジェクト内の第1のものの中に形成することを含む、インストラクションセットの種々のバージョンのプロセス(複数)をインタープリットする方法。

【請求項46】 複数のコンピュータを有するコンピュータネットワークにおいて、

各々がゼロ又はそれ以上のエージェントプロセスのための前記コンピュータ内の1つのもののある場所である複数のブレイスプロセスを前記コンピュータネットワーク中に用意し、

前記複数のブレイスプロセス中の目的点ブレイスプロセスへのエージェントプロセスのトリップをチケット手段によって指定し、

前記エージェントプロセス中の send オペレーションに応答して前記エージェントプロセスの1つのクローンを前記目的点ブレイスプロセスに転送することを特徴とする通信プロセス。

【請求項47】 複数のコンピュータを有するコンピュータネットワークにおいて、

前記複数のブレイスプロセス中の第2の目的点ブレイスプロセスへの前記第2のエージェントプロセスの前記トリップと異なる第2のトリップを第2のチケット手段によって指定することを特徴とする請求項第46項記載の通信プロセス。

【請求項48】 複数のコンピュータを有するコンピュータネットワークにおいて、前記第1のトリップを行な

う前記クローンと前記第2のトリップを行なう第2のクローンとがどちらも前記コンピュータの内の単一のコンピュータに転送されるべきことを定め、

前記第1のクローンを前記単一のコンピュータに移動させ、前記単一のコンピュータ中に、前記第1のクローンと異なる前記第2のクローンを形成することを特徴とする請求項第47項記載の通信プロセス。

【請求項49】 複数のコンピュータを有するコンピュータネットワークにおいて、前記第1のクローンを前記目的点ブレイスプロセスに転送することを特徴とする請求項第48項記載の通信プロセス。

【請求項50】 複数のコンピュータを有するコンピュータネットワークにおいて、前記第2のクローンを前記第2の目的点ブレイスプロセスに転送することを特徴とする請求項第48項記載の通信プロセス。

【請求項51】 複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェントプロセス中の go オペレーションに応答して、前記チケット手段によって指定される前記目的点ブレイスプロセスに前記エージェントプロセスを転送することを特徴とする請求項第46項記載の通信プロセス。

【請求項52】 複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段中のネームによって前記目的点ブレイスプロセスを指定することを特徴とする請求項第46項記載の通信プロセス。

【請求項53】 複数のコンピュータを有するコンピュータネットワークにおいて、前記ネームがテレネームであることを特徴とする請求項第52項記載の通信プロセス。

【請求項54】 複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段中のアドレスによって前記目的点ブレイスプロセスを指定することを特徴とする請求項第46項記載の通信プロセス。

【請求項55】 複数のコンピュータを有するコンピュータネットワークにおいて、前記アドレスがテレアドレスであることを特徴とする請求項第54項記載の通信プロセス。

【請求項56】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスがその1つのメンバであるクラスのサイテーションによって前記目的点ブレイスプロセスを前記チケット手段中において指定することを特徴とする請求項第46項記載の通信プロセス。

【請求項57】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスのアドレス、前記目的点ブレイスプロセスのネーム及び前記目的点ブレイスプロセスがその1つのメンバであるクラスのサイテーションの任意の組み合わせによって前記目的点ブレイスプロセスを前記チケット手段中において指定することを特徴とする請求項第46項記載の通

信プロセス。

【請求項58】 複数のコンピュータを有するコンピュータネットワークにおいて、前記トリップの最大の期間を前記チケット手段中に指定することを特徴とする請求項第46項記載の通信プロセス。

【請求項59】 複数のコンピュータを有するコンピュータネットワークにおいて、前記トリップのための望ましい期間を前記チケット手段中に指定することを特徴とする請求項第46項記載の通信プロセス。

【請求項60】 複数のコンピュータを有するコンピュータネットワークにおいて、転送手段を前記チケット手段中に指定し、該転送手段は前記クローンを転送するためのコンピュータ間通信手段の形式を指定することを特徴とする請求項第46項記載の通信プロセス。

【請求項61】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記エージェントプロセスのデッドラインを、前記チケット手段中に含まれる目的点パーミットによって制御することを特徴とする請求項第46項記載の通信プロセス。

【請求項62】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記エージェントプロセスが遂行することが許可されたオペレーションを、前記チケット手段に含まれる目的点パーミットによって制御することを特徴とする請求項第46項記載の通信プロセス。

【請求項63】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記エージェントプロセスが消費することが許可されたリソースを、前記チケット手段に含まれる目的点パーミットによって制御することを特徴とする請求項第46項記載の通信プロセス。

【請求項64】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいての他のプロセスに対する前記目的点ブレイスプロセスにおいての前記エージェントプロセスの相対的なプライオリティを、前記チケット手段に含まれる目的点パーミットによって指定することを特徴とする請求項第46項記載の通信プロセス。

【請求項65】 複数のコンピュータを有するコンピュータネットワークにおいて、あるプロセスが持つことを許可される能力をパーミット手段を介して制御することを特徴とする請求項第46項記載の通信プロセス。

【請求項66】 複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段が固有のパーミットであることを特徴とする請求項第65項記載の通信プロセス。

【請求項67】 複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段がローカルパーミットであることを特徴とする請求項第66項

記載の通信プロセス。

【請求項68】 複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段が一時的なパーミットであることを特徴とする請求項第65項記載の通信プロセス。

【請求項69】 複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が前記プロセスの消費するリソースからなることを特徴とする請求項第65項記載の通信プロセス。

【請求項70】 複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、その後ではプロセスが進行しえないデッドラインを含むことを特徴とする請求項第65項記載の通信プロセス。

【請求項71】 複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が他のプロセスに対する前記プロセスの相対的なプライオリティを含むことを特徴とする請求項第65項記載の通信プロセス。

【請求項72】 複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、前記プロセスが選択されたオペレーションを遂行する許可を含むことを特徴とする請求項第65項記載の通信プロセス。

【請求項73】 複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェントプロセスがオブジェクトを所有することを特徴とする請求項第46項記載の通信プロセス。

【請求項74】 複数のコンピュータを有するコンピュータネットワークにおいて、前記オブジェクトがダイジェストを有することを特徴とする請求項第73項記載の通信プロセス。

【請求項75】 複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェントプロセスのクローンを転送する前記工程が前記オブジェクトのコピーを転送することを含むことを特徴とする請求項第74項記載の通信プロセス。

【請求項76】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記コピーを第2のオブジェクトに相互変換し、

該第2のオブジェクトは、最初に述べた前記ダイジェストと同等のダイジェストを有し、

前記コピーは前記目的点ブレイスプロセスには転送されないことを特徴とする請求項第75項記載の通信プロセス。

【請求項77】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記オブジェクトの代わりに、相互変換されたオブジェクトを使用して、前記オブジェクトを前記目的点ブレイスプロセスに転送するのを除くことを特徴とする請求項第73項記載の通信プロセス。

【請求項78】 複数のコンピュータを有するコンピュ

ータネットワークにおいて、前記転送工程が前記目的点ブレイスプロセスにエンタする工程を含むことを特徴とする請求項第46項記載の通信プロセス。

【請求項79】 複数のコンピュータを有するコンピュータネットワークにおいて、前記転送ステップが前記ソースブレイスプロセスをエクシットする工程を含むことを特徴とする請求項第46項記載の通信プロセス。

【請求項80】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスがミーティングブレイスプロセスであり、前記ミーティングブレイスはリクエストエージェントプロセスとペティションドエージェントプロセスとの間のミーティングを取り計らうことを特徴とする請求項第46項記載の通信プロセス。

【請求項81】 コンピュータにおいて、第1のエージェントプロセスと第2のエージェントプロセスとを用意し、

ペティション手段によって、前記第1のエージェントプロセスと第2のエージェントプロセスとの間のミーティングを指定し、

前記ペティション手段によって定義された前記第1のエージェントプロセスと第2のエージェントプロセスとの間の前記ミーティングを取り計らうことを特徴とする通信プロセス。

【請求項82】 コンピュータにおいて、前記第1のエージェントプロセス中に前記ペティション手段を形成することを特徴とする請求項第81項記載の通信プロセス。

【請求項83】 コンピュータにおいて、ミーティングを特定する前記工程は、前記ペティション手段中において前記第2のエージェントプロセスを指定することを特徴とする請求項第81項記載の通信プロセス。

【請求項84】 コンピュータにおいて、前記第2のエージェントプロセスを指定する前記工程が、前記第2のエージェントをネームによって前記ペティション手段中において指定することを特徴とする請求項第83項記載の通信プロセス。

【請求項85】 コンピュータにおいて、前記ネームがテレネームであることを特徴とする請求項第84項記載の通信プロセス。

【請求項86】 コンピュータにおいて、前記第2のエージェントプロセスを指定する前記工程が、前記第2のエージェントプロセスがその1つのメンバであるクラスを前記ペティション手段中において指定することを特徴とする請求項第83項記載の通信プロセス。

【請求項87】 コンピュータにおいて、前記第2のエージェントプロセスを指定する前記工程は、前記第2のエージェントプロセスがその1つのメンバであるクラスのサイテーションを前記ペティション手段中において指定することを特徴とする請求項第83項記載の通信プ

ロセス。

【請求項88】 コンピュータにおいて、ミーティングを指定する前記工程が、前記ミーティングを取り計らうための最大の期間を前記ベティション手段中において指定することを特徴とする請求項第81項記載の通信プロセス。

【請求項89】 コンピュータにおいて、前記ミーティングを取り計らう前記工程は、前記第2のエージェントプロセスへのリファレンスを前記第1のエージェントプロセスに供与することを特徴とする請求項第81項記載の通信プロセス。

【請求項90】 コンピュータにおいて、前記ミーティングを取り計らう前記工程が、前記第1のエージェントプロセスへのリファレンスを前記第2のエージェントプロセスに対して供与することを特徴とする請求項第89項記載の通信プロセス。

【請求項91】 コンピュータにおいて、前記第2のエージェントプロセス中のオペレーションの遂行を前記第1のエージェントプロセス中においてインストラクションに应答して生じさせることを特徴とする請求項第81項記載の通信プロセス。

【請求項92】 コンピュータにおいて、前記第1のエージェントプロセスがオブジェクトを所有することを特徴とする請求項第91項記載の通信プロセス。

【請求項93】 コンピュータにおいて、前記オブジェクトへのリファレンスを前記第2のエージェントプロセスに対して供与することを特徴とする請求項第92項記載の通信プロセス。

【請求項94】 コンピュータにおいて、前記リファレンスが保護されるリファレンスであることを特徴とする請求項第93項記載の通信プロセス。

【請求項95】 コンピュータにおいて、前記オブジェクトのコピーへのリファレンスを前記第2のエージェントプロセスに供与することを特徴とする請求項第92項記載の通信プロセス。

【請求項96】 コンピュータにおいて、前記第2のエージェントプロセスがオブジェクトを所有することを特徴とする請求項第91項記載の通信プロセス。

【請求項97】 コンピュータにおいて、前記オブジェクトへのリファレンスを前記第1のエージェントプロセスに対して供与することを特徴とする請求項第96項記載の通信プロセス。

【請求項98】 コンピュータにおいて、前記リファレンスがプロテクトされたリファレンスであることを特徴とする請求項第97項記載の通信プロセス。

【請求項99】 コンピュータにおいて、前記オブジェクトのコピーへのリファレンスを前記第1のエージェントプロセスに対して供与することを特徴とする請求項第96項記載の通信プロセス。

【請求項100】 複数のコンピュータを有するコンピ

ュータネットワークにおいて、チケット手段及び send オペレーションを有するエージェント手段と、前記複数のコンピュータの1つにおいてその各々が動作する複数のプレイス手段とを有し、

前記エージェント手段は前記複数のプレイス手段中の第1のプレイス手段であり、

前記チケット手段は、前記複数のプレイス手段中の目的点プレイス手段への前記エージェント手段のトリップを指定し、

前記 send オペレーションは、前記エージェント手段の1つのクローンを前記目的点手段に転送することを特徴とする通信システム。

【請求項101】 複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段が、前記エージェント手段のための第2の目的点プレイス手段への第2のトリップを指定する、前記チケット手段と異なる第2のチケット手段を、含むことを特徴とする請求項第100項記載の通信システム。

【請求項102】 複数のコンピュータを有するコンピュータネットワークにおいて、前記 send オペレーションの遂行によって前記エージェント手段の第2のクローンが前記第2の目的点プレイス手段に転送されることを特徴とする請求項第101項記載の通信システム。

【請求項103】 複数のコンピュータを有するコンピュータネットワークにおいて、前記トリップ及び前記第2のトリップが、前記複数のコンピュータ中のある単一のコンピュータへの転送を含み、

前記クローンを前記目的点プレイス手段に転送し、前記第2のクローンを前記第2の目的点手段に転送する際に、前記 send オペレーションの遂行によって、(a) 前記第1のクローンが前記単一のコンピュータに転送され、(b) 前記単一のコンピュータ中において前記第1のクローンから前記第2のクローンが形成されることを特徴とする請求項第102項記載の通信システム。

【請求項104】 複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段が go オペレーションを含み、

前記 go オペレーションは、第3のチケット手段によって指定された第3の目的点プレイス手段に前記エージェント手段を転送することを特徴とする請求項第100項記載の通信システム。

【請求項105】 複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記目的点プレイス手段を特定するネームを含むことを特徴とする請求項第100項記載の通信システム。

【請求項106】 複数のコンピュータを有するコンピュータネットワークにおいて、前記ネームがテレネームであることを特徴とする請求項第105項記載の通信システム。

【請求項107】 複数のコンピュータを有するコンピ

ユータネットワークにおいて、前記チケット手段が前記目的点プレイス手段のためのアドレスを含むことを特徴とする請求項第100項記載の通信システム。

【請求項108】 複数のコンピュータを有するコンピュータネットワークにおいて、前記アドレスがテレアドレスであることを特徴とする請求項第107項記載の通信システム。

【請求項109】 複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記目的点プレイス手段をその1つのメンバとするクラスを指定するサイテーション手段を含むことを特徴とする請求項第100項記載の通信システム。

【請求項110】 複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が、前記目的点プレイス手段のアドレスと、前記目的点プレイス手段のネームと、前記目的点プレイス手段をその1つのメンバとするクラスを指定するサイテーション手段とからなる群中より選ばれた任意の組合せを含むことを特徴とする請求項第100項記載の通信システム。

【請求項111】 複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記トリップのための最大の期間を指定する手段を含むことを特徴とする請求項第100項記載の通信システム。

【請求項112】 複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記トリップのための望ましい期間を指定する手段を含むことを特徴とする請求項第100項記載の通信システム。

【請求項113】 複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が、前記トリップ手段を完成させるコンピュータ間通信手段の形式を指定するウェイ手段を含むことを特徴とする請求項第100項記載の通信システム。

【請求項114】 複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が、前記目的点プレイス手段においての前記エージェント手段のためのデッドラインを制御するパーミット手段を含むことを特徴とする請求項第100項記載の通信システム。

【請求項115】 複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段が前記目的点プレイス手段において遂行することが許可されているオペレーションを制御するためのパーミット手段を前記チケット手段を有することを特徴とする請求項第100項記載の通信システム。

【請求項116】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点プレイス手段において前記エージェント手段が消費することが許可されているリソースを制御するためのパーミット手段を前記チケット手段が有することを特徴とする請求項第100項記載の通信システム。

【請求項117】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点プレイス手段においての他のエージェント手段に対する前記目的点プレイス手段においての前記エージェント手段の相対的なプライオリティを指定するためのパーミット手段を前記チケット手段が有することを特徴とする請求項第100項記載の通信システム。

【請求項118】 複数のコンピュータを有するコンピュータネットワークにおいて、エージェント手段が持つことが許可される能力を制御するためのパーミット手段を有することを特徴とする請求項第100項記載の通信システム。

【請求項119】 複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段が固有のパーミットであることを特徴とする請求項第118項記載の通信システム。

【請求項120】 複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段がローカルパーミットであることを特徴とする請求項第118項記載の通信システム。

【請求項121】 複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段が一時的なパーミットであることを特徴とする請求項第118項記載の通信システム。

【請求項122】 複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が前記エージェント手段の消費するリソースを含むことを特徴とする請求項第118項記載の通信システム。

【請求項123】 複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、前記エージェント手段がそれから後進行することのできないデッドラインを含むことを特徴とする請求項第118項記載の通信システム。

【請求項124】 複数のコンピュータを有するコンピュータネットワークにおいて、他のエージェント手段を含み、前記能力は、前記他のエージェント手段に対する前記エージェント手段の相対的なプライオリティを含むことを特徴とする請求項第118項記載の通信システム。

【請求項125】 複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、前記エージェント手段が選択されたオペレーションを遂行する許可を含むことを特徴とする請求項第118項記載の通信システム。

【請求項126】 複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段がオブジェクトを所有することを特徴とする請求項第100項記載の通信システム。

【請求項127】 複数のコンピュータを有するコンピュータネットワークにおいて、前記オブジェクトがダイ



ジェストを有することを特徴とする請求項第126項記載の通信システム。

【請求項128】 複数のコンピュータを有するコンピュータネットワークにおいて、前記オブジェクトのコピーが前記クローンとともに、前記 send オペレーションの遂行によって転送されることを特徴とする請求項第127項記載の通信システム。

【請求項129】 複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイス手段において第2のオブジェクトは前記コピーに相互変換され、

前記第2のオブジェクトは、前記ダイジェストと同等のダイジェストを持つことによって前記コピーを前記目的点ブレイス手段に転送することを不要とすることを特徴とする請求項第128項記載の通信システム。

【請求項130】 複数のコンピュータを有するコンピュータネットワークにおいて、前記 send オペレーションが前記目的点ブレイス手段においての前記コピーの代わりに相互変換されたオブジェクトを使用することによって、前記オブジェクトの前記コピーを前記目的点ブレイス手段に転送する必要を除くことを特徴とする請求項第128項記載の通信システム。

【請求項131】 複数のコンピュータを有するコンピュータネットワークにおいて、前記ブレイス手段は前記ブレイス手段にエンタするための手段を有することを特徴とする請求項第100項記載の通信システム。

【請求項132】 複数のコンピュータを有するコンピュータネットワークにおいて、前記ブレイス手段が前記ブレイス手段をエグジットする手段を有することを特徴とする請求項第100項記載の通信システム。

【請求項133】 複数のコンピュータを有するコンピュータネットワークにおいて、前記ブレイス手段がミーティングブレイス手段を含み、

前記ミーティングブレイス手段はリクエストエージェント手段とレスポンスエージェント手段との間のミーティングを取り計らうことを特徴とする請求項第100項記載の通信システム。

【請求項134】 コンピュータにおいて、meet オペレーションを有するミーティングブレイス手段を有し、該ミーティングブレイスは、複数のエージェント手段のための場所であり、更に、

ペティション手段と、

前記複数のエージェント手段の内の第1のエージェント手段と、

前記複数のエージェント手段の中の第2のエージェント手段とを有し、

前記第1のエージェント手段及び第2のエージェント手段は前記複数のエージェント手段の内のどれともミートでき、

前記ペティション手段は前記第1のエージェント手段

と第2のエージェント手段との間のミーティングを指定し、

前記ミーティング手段は前記 meet オペレーションに回答して前記ミーティングを取り計らうことを特徴とする通信システム。

【請求項135】 コンピュータにおいて、前記第1のエージェント手段が前記ペティション手段を有することを特徴とする請求項第134項記載の通信システム。

【請求項136】 コンピュータにおいて、前記ペティション手段が前記第2のエージェントプロセスを規定することによって前記ミーティングを指定することを特徴とする請求項第135項記載の通信システム。

【請求項137】 コンピュータにおいて、前記ペティション手段が前記第2のエージェント手段を指定するネームを含むことを特徴とする請求項第136項記載の通信システム。

【請求項138】 コンピュータにおいて、前記ネームがテレネームであることを特徴とする請求項第137項記載の通信システム。

【請求項139】 コンピュータにおいて、前記第2のエージェントプロセスがその1つのメンバであるクラスを指定することによって前記ペティション手段が前記第2のエージェントを指定することを特徴とする請求項第136項記載の通信システム。

【請求項140】 コンピュータにおいて、前記ペティション手段がサイテーションを含み、

該サイテーションは、前記第2のエージェントプロセスがその1つのメンバであるクラスを指定することを特徴とする請求項第136項記載の通信システム。

【請求項141】 コンピュータにおいて、前記ペティション手段が前記ミーティングを取り計らうための最大の期間を規定することを特徴とする請求項第134項記載の通信システム。

【請求項142】 コンピュータにおいて、前記 meet オペレーションが前記第2のエージェント手段へのリファレンスを前記第1のエージェント手段に供与することを特徴とする請求項第134項記載の通信システム。

【請求項143】 コンピュータにおいて、前記 meet オペレーションが前記第1のエージェント手段へのリファレンスを前記第2のエージェント手段に供与することを特徴とする請求項第142項記載の通信システム。

【請求項144】 コンピュータにおいて、前記第1のエージェント手段が前記第2のエージェント手段中のオペレーションの遂行を生じさせることを特徴とする請求項第134項記載の通信システム。

【請求項145】 コンピュータにおいて、前記第1のエージェント手段がオブジェクトを所有することを特徴とする請求項第144項記載の通信システム。

【請求項146】 コンピュータにおいて、前記第1のエージェント手段が前記オブジェクトへのリファレンス

を前記第2のエージェント手段に供与することを特徴とする請求項第145項記載の通信システム。

【請求項147】 コンピュータにおいて、前記リファレンスがプロテクトされたリファレンスであることを特徴とする請求項第146項記載の通信システム。

【請求項148】 コンピュータにおいて、前記第1のエージェント手段が前記オブジェクトのコピーへのリファレンスを前記第2のエージェント手段に供与することを特徴とする請求項第145項記載の通信システム。

【請求項149】 コンピュータにおいて、前記第2のエージェント手段がオブジェクトを所有することを特徴とする請求項第144項記載の通信システム。

【請求項150】 コンピュータにおいて、前記第2のエージェント手段が前記オブジェクトへのリファレンスを前記第1のエージェント手段に供与することを特徴とする請求項第149項記載の通信システム。

【請求項151】 コンピュータにおいて、前記リファレンスがプロテクトされたリファレンスであることを特徴とする請求項第150項記載の通信システム。

【請求項152】 コンピュータにおいて、前記第2のエージェント手段が前記オブジェクトのコピーへのリファレンスを前記第1のエージェント手段に供与することを特徴とする請求項第149項記載の通信システム。

【請求項153】 1以上のコンピュータを有するコンピュータネットワークにおいて、第1のエンジンプロセスから第2のエンジンプロセスにデータを移送するデータ移送方法において、

(a) 前記第1のエンジンプロセス及び第2のエンジンプロセスを含む1以上のエージェントプロセスを実行するための実行手段を用意し、各々の前記エージェントプロセスは、コンピュータインストラクションセットからの複数のインストラクションを含み、実行状態を持つものとし、

(b) 前記コンピュータインストラクションセット中に go インストラクションを用意し、該 go インストラクションは、前記第1のエンジンプロセス中において実行される第1のエージェントプロセスに含まれるものとし、前記 go インストラクションの遂行によって、

(i) 前記第1のエンジンプロセスによる前記第1のエージェントプロセスの実行の中断と、(i i) 前記第1のエージェントプロセスの前記実行状態が保存されるような前記第1のエージェントプロセスの表現と、(i i i) 前記第1のエンジンプロセスから第2のエンジンプロセスへの前記第1のエージェントプロセスの前記表現の移送と、(i v) 前記第2のエンジンプロセスによる前記第1のエージェントプロセスの実行の再開とを生じさせ、

(c) 前記第1のエージェントプロセスによる前記エージェントプロセスの実行を生じさせることによって前記 go インストラクションの遂行を生じさせることを特徴

とするデータ移送方法。

【請求項154】 前記コンピュータインストラクションセットがオブジェクト指向であることを特徴とする請求項第153項記載のデータ移送方法。

【請求項155】 前記複数のエージェントプロセスが前記オブジェクト指向コンピュータインストラクションセットのオブジェクトであることを特徴とする請求項第154項記載のデータ移送方法。

【請求項156】 前記 go インストラクションの遂行を生じさせる前に前記第1のエージェントプロセスにデータを付加し、

前記第1のエージェントプロセスの前記表現が前記データを含み、

前記第2のエンジンプロセスへの前記第1のエージェントプロセスの前記表現の移送が前記データを移送を含むことを特徴とする請求項第153項記載のデータ移送方法。

【請求項157】 前記第1のエージェントプロセスが、前記第1のエンジンプロセスから前記第2のエンジンプロセスに転送されるべきメッセージを表すデータを含み、

前記 go インストラクションの実行によって生ずる前記第1のエージェントプロセスの移送が前記第1のエンジンプロセスから前記第2のエンジンプロセスへの前記データの移送を生じさせることを特徴とする請求項第153項記載のデータ移送方法。

【請求項158】 前記第1のエンジンプロセスが、第1のコンピュータにおいて実行され、前記第2のエンジンプロセスが第2のコンピュータによって実行され、前記第1及び第2のコンピュータが前記コンピュータネットワークの一部分であることを特徴とする請求項第153項記載のデータ移送方法。

【請求項159】 第1のエンジンプロセスから1以上のエンジンプロセスにデータを移送するデータ移送方法において、

(a) コンピュータインストラクションセットからのインストラクションを含む複数のエージェントプロセスを実行するための手段を用意し、該エージェントプロセスを実行する手段が、前記第1のエンジンプロセス及び1以上のエンジンプロセスを含み、各々の該エージェントプロセスは実行状態を含み、

(b) 前記コンピュータインストラクションセット中に send インストラクションを用意し、該 send インストラクションは、前記複数のエージェントプロセス中の第1のエージェントプロセス中に含まれるようにし、前記 send インストラクションの実行は、(i) 前記第1のエージェントプロセスの1以上のコピーを形成し、これらのコピーが、前記第1のエージェントプロセスの前記実行状態を保存するとともにそれを含み、(i i) 前記第1のエージェントプロセスのコピーの各々を前記第1のエン

ジンプロセスから前記１以上のエンジンプロセスのそれぞれに移送し、（iii）前記第１のエージェントプロセスの前記コピーの各々を前記１以上のエンジンプロセスのそれぞれの１つによって実行して前記第１のエージェントプロセスの再開された実行をシミュレートすることを特徴とするデータ移送方法。

【請求項１６０】 前記第１のエージェントプロセスの前記コピーの２以上が前記第１のエンジンプロセスから単一の第２のエンジンプロセスに移送されるべき条件のもとにおいて、前記第１のエージェントプロセスの前記２以上のコピーの移送が、前記第１のエージェントプロセスの単一のコピーを前記第２のエンジンプロセスに移送する工程と、前記第２のエンジンプロセス中において前記単一のコピーから、前記第１のエージェントプロセスの前記２以上のコピーを形成する工程とからなることを特徴とする請求項第１５９項記載のデータ移送方法。

【請求項１６１】 あるコンピュータシステムにおいて実行されている第１のエージェントプロセスから、該コンピュータシステムによって実行されている第２のエージェントプロセスにデータを転送する際に、前記第１のエージェントプロセス及び第２のエージェントプロセスがあるプレイスプロセスの占有者であるようにしてデータを転送するデータ転送方法において、前記第１のエージェントプロセスが送出するミートインストラクションに応答して、前記第２のエージェントプロセスによるプロシージャの実行を生じさせる工程を含み、前記プロシージャは、前記第２のエージェントプロセスの一部であり、前記第２のエージェントコンピュータプロセスに含まれるコンピュータインストラクションのコレクションを含み、前記第２のエージェントプロセスによって送出される、前記プロシージャの一部である第２のインストラクションに応答して、前記第２のエージェントプロセスへのアクセス手段を前記第１のエージェントプロセスに供与する工程を含むことを特徴とするデータ転送方法。

【請求項１６２】 前記第２のエージェントプロセスによるプロシージャの実行を生じさせる前記工程は、前記第１のエージェントプロセスによって送出される前記ミートインストラクションに応答して前記プレイスプロセスによる第２のプロシージャの実行を生じさせることを含み、前記プレイスプロセスによる前記第２のプロシージャの実行が、前記プロシージャを前記第２のエージェントプロセスによって実行させるインストラクションを送出することを特徴とする請求項第１６１項記載のデータ転送方法。

【請求項１６３】 前記第２のエージェントプロセスへのアクセス手段が前記プレイスプロセスによって前記第１のエージェントプロセスに供与されることを特徴とする請求項第１６１項記載のデータ転送方法。

【請求項１６４】 近似的に、前記プレイスプロセスが前記第２のエージェントプロセスへの前記アクセス手段を前記第１のエージェントプロセスに供与する時点において、前記プレイスプロセスが前記第１のエージェントプロセスへのアクセス手段を前記第２のエージェントプロセスに供与することを特徴とする請求項第１６３項記載のデータ転送方法。

【請求項１６５】 第１のＣＰＵ及び第１のメモリを含む第１のコンピュータシステムから、第２のＣＰＵ及び第２のメモリを含む第２のコンピュータシステムへ第１のコンピュータプロセスを移送するプロセスの移送方法において、ある実行状態を持つ前記第１のコンピュータプロセスの実行を前記第１のＣＰＵにおいて開始し、前記第１のＣＰＵ内において前記第１のコンピュータプロセスの実行を中断し、前記第１のコンピュータプロセスをデータとして前記第１のメモリ中に表現し、前記データは前記第１のコンピュータプロセスの実行が中断された時点においての前記第１のコンピュータプロセスの前記実行状態を含むものとし、前記データを前記第１のメモリから前記第２のメモリに移送し、前記データ中に表現された前記実行状態を有する第２のコンピュータプロセスを前記データから前記第２のコンピュータシステム上に形成し、前記第２のコンピュータプロセスを実行することによって前記第２のＣＰＵ内において前記第１のコンピュータシステムの実行の再開を実効的にシミュレートすることを特徴とするプロセスの移送方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、リモートプログラミングの実施方法に関し、特に分散型計算環境であって、オブジェクト指向環境において複数のプロセスがつくり出され、コンピュータネットワークを通じてそれ自身の運動を差し向けるようにしたリモートプログラミングの実施方法等に関する。

【０００２】

【従来の技術】今日多くの異なった種類のコンピュータシステムが利用されているが、これらのシステムの多くは、図１に示すように、基本的なコンポーネントから構成されている。典型的には、コンピュータシステム２０は中央処理ユニット装置（以下、ＣＰＵという）を備え、このＣＰＵ１０は、入力バス１１を介して入力モジュール１２に接続されるとともに、出力バス１３を経て出力モジュール１４に接続されている。また、ＣＰＵ１０は、データバス１５、１６を経てメモリユニット１７にも接続されている。

【０００３】ＣＰＵ１０は、制御及び計算機能を有す

る。入力モジュール12及び出力モジュール14は、コンピュータのユーザーとCPU10間の通信のために用いられる。入力モジュール12は、情報をCPU10に供給する。典型的な入力モジュールはキーボード及びマウスである。出力モジュール14は、CPU10からの情報を表示する。典型的な出力モジュールはビデオディスプレイモニタ、プリンタ、プロッタ及びその他の視覚表示手段である。入力モジュール12及び出力モジュール14は、ときには、インプット／アウトプット（I/O）ユニットとも呼ばれる。

【0004】メモリユニット17は、典型的には2種の一般的な形式の情報、すなわちインストラクション（命令）とデータを収容している。コンピュータプログラムは、ある特定の機能を実現するためにコンピュータによって実行される一連のインストラクションである。メモリユニット17のデータは、CPU10において実行されているコンピュータプログラムからのインストラクションにตอบสนองして、CPU10によって処理される。実行中のコンピュータプログラムは、コンピュータプロセス（computer process）と呼ばれる。

【0005】メモリユニット17は、典型的には、マスメモリ17A（ときには二次メモリとも呼ばれる）と、メインメモリ17Bとを備えている。メインメモリ17Bは、CPU10によって現に実行されているプログラムの少なくとも一部分と、そのプログラムによって必要とされるデータを記憶するために（通常）用いられる。

マスメモリ17A、例えば磁気ディスクや磁気テープは、プログラム、データ、又はCPU10によって直には必要とされないか若しくはメインメモリ17Bの大きさの制限のためにメインメモリ17B中に収容することのできないプログラム又はデータの一部分を記憶するために用いられる。

【0006】コンピュータのオペレーティングシステムは、CPU10、入力モジュール12、出力モジュール14、マスメモリ17A、メインメモリ17Bを連動させ、ユーザーアプリケーション（以下に定義する）とコンピュータハードウェア間のインターフェイスを行なうコンピュータプロセスである。ここで使用されているコンピュータハードウェアという用語は、コンピュータシステムの物理的なコンポーネントを意味する。コンピュータシステムを正しく動作させるためにはオペレーティングシステムが必要であるから、オペレーティングシステムは、コンピュータシステムのパワーアップ時にメインメモリ17Bにロードされ（ブーティングと呼ばれるプロセス）、コンピュータシステムが最終的に不動作にされるまでメインメモリ17Bに留められて実行される。

【0007】ユーザーアプリケーションは、一般に、オペレーティングシステムとは別に実行され、オペレーティングシステムとは別のコンピュータプロセスである。

ユーザーアプリケーションは、ソースコードすなわち人にわかりやすい形のコンピュータインストラクションとして時間中のある時点に置いて実行される。このソースコードは、コンピュータシステム20、特にCPU10によって理解される形のオブジェクトコードに翻訳され、すなわちコンパイルされる。いくつかのオブジェクトコードモジュールは、コンピュータシステム20のCPU10によって実行可能な単一のイメージとなるようにリンクされる。プログラムは、この実行可能なイメージをメインメモリ17Bにコピーし、この実行可能なイメージのインストラクションを1つずつ実行するようにCPU10に指示することにより、実行される。前述したように、このような実行状態にあるコンピュータプログラムは、プロセス（process）と呼ばれる。

【0008】コンピュータシステム20で実行されているプロセスのインストラクションは、一般に、コンピュータシステムのリソース（資源）を操作（マニピュレート）する。例えば、これらのインストラクションは、マスメモリ17Aに記憶されているデータを操作し、入力モジュール12からデータを受け入れ、又は出力モジュール14にデータを表示することができる。第1のコンピュータシステムにおいて実行されているあるプロセスは、第2のコンピュータシステムのリソースを操作することができるコンピュータインストラクションを発生させるように構成されている。

【0009】2以上のコンピュータシステム間でデータを転送することができるように、2以上のコンピュータシステムを接続することは、（コンピュータ通信技術において、）周知である。このように接続された2以上のコンピュータシステムはネットワークと呼ばれる。このネットワークにおいて、各々のコンピュータシステムは、データの送受信を行なう入力／出力装置として（他のコンピュータシステムを）取り扱う。

【0010】第1のコンピュータシステムにおいて実行されている第1のプロセスは、第2のコンピュータシステムにおいて実行されている第2のプロセスにインストラクションを送出することができる。その場合、第2のプロセスは、第1のプロセスから受取ったインストラクションに従って第2のコンピュータシステムのリソースを操作する。第2のプロセスは、クライアント（client）プロセスと呼ばれる第1のプロセスにサービスを提供するものであるため、サーバー（server）プロセスと呼ばれる。この提供されるサービスは、第2のコンピュータシステムのリソースのマニピュレーションである。

【0011】多くのコンピュータ通信システムでは、リモートプロシージャコーリング（remote procedure calling）を実行する。リモートプロシージャコーリングにおいては、クライアントプロセス（第1のコンピュータシステムにおいて実行されており、第2のコンピュータシステムのデータを操作することを求める）は、サーバ

ープロセス（第2のコンピュータシステムにおいて実行され、クライアントプロセスからの要求に回答してデータを操作する）にアクションを要求する。サーバープロセスは、クライアントプロセスのために、不可避免的に有限なオペレーションリストのどれかを遂行する。クライアントプロセスによって要求された正確な機能が、サーバープロセスによって遂行される特別な操作の中に含まれないことが（しばしば）ある。サーバープロセスの定義されていない機能を実行するには、サーバープロセスへのいくつかの要求及びサーバープロセスからの回答を必要とし、これは通信メディアの実質的な使用を必要とし、それによってコスト及び時間が非常に増大する。

【0012】図2に示すフローチャート（ロジックフローダイアグラム）は、リモートプロシージャコーリングの具体例を示している。このフローチャートは、図3に示すネットワーク256を介して相互接続されたコンピュータシステム20A、20Bのコンテキスト（context）によって説明される。ある与えられたパターン例えば“TMP”に終わるファイル名に名前が一致し、少なくとも30日の間変更されなかったコンピュータシステム20B上の全てのファイルを、コンピュータシステム20A上のクライアントプロセス252が削除するものと想定する。クライアントプロセス252は、コンピュータシステム20Bにおいて実行されているサーバープロセス254に、ステップ21において、全てのファイルをリストする要求を送出する図2に示す。矢印260

（図3）はこの要求を表わしている。処理はステップ21（図2）からステップ22に進み、このステップ22において、クライアントプロセス252（図3）は、矢印262に示すように、サーバープロセス254からそのリストを受け取る。ステップ2122（図2）での処理はネットワークの通信メディアを介しての2回の情報転送（トランスファ）を含む。

【0013】図2において、2重の矩形は、ネットワーク256（図3）を横断する情報転送を表している。コンピュータ20A又はコンピュータ20B内においてのインストラクションの実行には数ナノ秒ないし数マイクロ秒を必要とするのに対し、ネットワーク256を介しての情報転送には、ネットワーク256の形態及び情報量に依存するが、一般に数秒から数分を必要とする。あるネットワークにおいては、大量のデータの転送に1時間以上もかかることがある。

【0014】処理は、ステップ22（図2）からステップ（FOR EACH FILENAME）23に移行する。ステップ23とステップ（NEXT）23a、23b、23cは、ファイル名リスト中の各ファイル名がクライアントプロセスによって処理されるループを形成している。リストの各ファイル名に対する処理は、ステップ23から判定ステップ24に進み、この判定ステップ24において、クライアントプロセス252（図3）は、ファイル名をパタ

ーンと比較する。もしファイル名がパターンと一致しない場合には、処理は、判定ステップ24（図2）からステップ23aを介してステップ23に戻り、その次のファイル名が処理される。

【0015】逆に、ファイル名がパターンに一致している場合には、処理は判定ステップ24からステップ25に進む。ステップ25において、クライアントプロセス252（図3）は、矢印264で示すように、そのファイル名が一致したファイルの最後に変更された日付を用意する旨のインストラクションをサーバープロセス254に送出する。処理は、ステップ25（図2）からステップ26に移行し、このステップ26において、クライアントプロセス252（図3）は、矢印266で示すように、要求した日付をサーバープロセス24から受取る。ステップ25、26（図2）は、それぞれこのような一致するファイル名についてのネットワークの通信メディアを介する2以上の転送を含む。

【0016】処理は、ステップ26から判定ステップ27に進み、この判定ステップ27において、クライアントプロセス252（図3）は、受信された日付と現在の日付から30日を引いた日付とを比較する。そして、ファイルが前の30日以内に更新されていたときは、処理は、判定ステップ27（図2）からステップ23bを介してステップ23に戻り、このステップ23において、次のファイル名が処理される。

【0017】逆に、30日以前にそのファイルが更新されていなかったとき、処理は、判定ステップ27からステップ28に進む。

【0018】ステップ28において、クライアントプロセス252（図3）は、矢印268で示すように、コンピュータシステム20B内からファイルを削除するインストラクションをサーバープロセス254に送出する。処理は、ステップ28からステップ29（図2）に進み、このステップ29において、クライアントプロセス252（図3）は、矢印270で示すように、その削除処理に対するアクノレジメント（acknowledgement）をサーバープロセス254から受ける。このように、ステップ28、29（図2）は、削除されるべき各ファイルについてのネットワーク256（図3）を介する2つの別の転送を含む。

【0019】処理は、ステップ29からステップ23cを介してステップ23に戻る。そしてリスト中の全てのファイル名が処理されたならば、処理は、ステップ23からステップ20dに進み、このステップ20bにおいて処理が終了する。この図2に示すフローチャートのよう処理には、ネットワーク256を介する多数の情報転送を必要とする。このようにネットワークの通信メディアを過度に使用することは時間及びコストの浪費になる。

【0020】リモートプロシージャコーリングの別のア

ブローチは、リモートプログラミング (remote programming) である。リモートプログラミングにおいて、クライアントプロセスと呼ばれる第1のコンピュータシステムにおいて実行されているプロセスは、第2のコンピュータシステムにおいて実行されているサーバプロセスに1つのインストラクションのリストを送出する。そして、これらのインストラクションは、サーバプロセスによって実行され、クライアントプロセスの目的が達せられる。サーバプロセスが実行するようにされているこれらのインストラクションは、ある程度の普遍性をもたねばならない。すなわち、これらのインストラクションはある程度の決定 (デシジョンメイキング) 能力を備えなければならない。

【0021】図4に示すフローチャートは、リモートプログラミングの具体例を示している。図2、図3において説明した具体例をリモートプログラミング環境において実現するには、クライアントプロセス352 (図5) は、ステップ31 (図4) においては、このステップ31の詳細なフローチャートに示すインストラクションから構成されてるプログラムを作成する。このステップ31の詳細について以下に説明する。

【0022】処理は、ステップ31からステップ32に進み、ステップ32において、プログラムは、矢印358で示すように、ネットワーク356を介しコンピュータ30B (図5) に転送される。処理は、ステップ32 (図4) からステップ33に進みこのステップ33において、プログラムはコンピュータ30Bによって実行される。この実行の中にはプログラムはコンピュータ30B上のプロセス352A (図5) となっている。そして、プロセス352Aにおいて、プログラムのインストラクションは、ステップ31のフローチャートに従って実行される。

【0023】ステップ31-B (図4) において、ファイル名のリストが作成される。処理は、ステップ31-Bからステップ (FOR EACH FILENAME) 31-Cに移行する。ステップ31-Cとステップ (NEXT) 31-H、31-I、31-Jは、ファイル名リスト中の各ファイル名を処理するループを形成している。各ファイル名について、ステップ31-Cから判定ステップ31-Dに進み、この判定ステップ31-Dにおいて、ファイル名がパターンと比較される。ファイル名がパターンと一致しないときは、処理は判定ステップ31-Dからステップ31-Cを介してステップ31-Cに戻る。逆に、ファイル名がパターンに一致するときは、処理は、判定ステップ31-Dからステップ31-Eに移行する。

【0024】ステップ31-Eにおいて、プロセス352Aは、一致したファイル名を有するファイルの最後の変更の日付けを検索する。処理は、ステップ31-Eから判定ステップ31-Fに進み、このステップ31-Fにおいて、日付が現在の日付から30日を引いたものと

比較される。

【0025】ファイルの最後の変更の日付は、サーバプロセス354 (図5) から検索される。最後の変更の日付が30日以前の日付よりも新しいときには、処理は、判定ステップ31-F (図4) からステップ31-Iを介してステップ31-Cに戻る。逆に、最後の変更の日付が30日以前の日付よりも更に以前であったときには、処理は、判定ステップ31-Fからステップ31-Gに進む。ステップ31-Gにおいて、サーバプロセス354 (図5) に適切なインストラクションを送出することによってファイルが削除される。処理はステップ31-G (図4) からステップ31-Jを介してステップ31-Cに戻る。リスト中の全てのファイル名が処理されたならば、処理は、ステップ31-Cからステップ31-Kに進み、ここでプロセス352Aが完了する。矢印360 (図5) に示すように、プロセス352Aとサーバプロセス354間の全ての相互作用は、ネットワーク356を使用することなく、全てコンピュータ30B中において起る。

【0026】プログラムが成功のうちに終了した後、処理は、ステップ33からステップ34 (図4) に移行し、このステップ34においてサーバプロセス354 (図5) は、矢印362によって示すようにプログラムが成功のうちに終了したことをクライアントプロセス352に報告する。このリモートプログラミングプロシージャがネットワーク通信メディアをただ2回しか使用していないことに注意されたい。第1回の使用は、矢印358によって示すようにインストラクションリスト又はプログラムをサーバプロセス354に送出する使用であり、第2の使用は、矢印362によって示すように、プログラムが成功のうちに終了したことの通知をサーバプロセス354から受ける使用である。サーバプロセス354の開発者によって明白に提供されていなかったり、おそらくは予想さえもされていなかったオペレーションを、クライアントプロセス352がサーバプロセス354に要求しうることにも注意されたい。

【0027】しかしリモートプログラミングは、2以上のコンピュータシステム上のデータの操作を連係化することをコンピュータプロセスが求める場合に、リモートプロシージャコーリングの場合と同様の非効率に遭遇する。一例としてコンピュータXのプロセスは、コンピュータXのどれかのファイルと同一の名前、最後の変更の日付及びサイズを有するコンピュータシステムY又はコンピュータシステムZの全てのファイルの削除を求めるかもしれない。図4、図5について前述したサーバプロセス354は、サーバプロセスが実行しているコンピュータシステムすなわちコンピュータシステムY上のリソースを直接操作することができるので、サーバプロセスは、遠隔プロセスコーリングに関連して前述した非効率を生ずることなしには、1以上のコンピュータシ

システムを横断するデータマネージメントを連係化することはできない。

【0028】遠隔プログラムに対する改良は、C. Daniel Wolfson, et al., "Intelligent Routers", 9th International Conference on Distributed Computing Systems, pp. 371-375 (1989) に記載されている。Wolfson 等は、あるネットワーク内においてあるプロセスが1つのコンピュータシステムから別のコンピュータシステムに移行することのできるシステムを開示している。このプロセスは、実行されたときにプロセスの移動を生じるインストラクションを送出することによってネットワークを通るそれ自身の運動を指示する。このインストラクションはそのプロセスが移動すべきコンピュータシステムを指定する。

【0029】しかしWolfson 等によって開示された形式による解決では、機能性が制限される。プロセス(複数)には、それぞれのプロセスが移動したコンピュータシステムの特別の要求に適合した挙動を示すように構成されることが要求される。プロセス相互が通信することはできない。その代わりにプロセスは、それぞれのプロセスが実行されているコンピュータシステムのマスメモリにデータを直接に記憶させたりこのマスメモリからデータを直接検索したりする。あるプロセスが別のプロセスと相互作用をしようのは、両方のプロセスがマスメモリ中においてメッセージを記憶させたり検索したりする正確なロケーションを知っている場合に限られる。

【0030】Wolfson 等によって開示されたシステムはいかなる安全性(セキュリティ)も与えない。すなわち、Wolfson 等のシステムがある1つのプロセスを別のプロセスと相互作用することを許容した限りにおいて、悪意のあるプロセスが第2プロセスの実行を妨げるかもしれない。更に、Wolfson 等などによって開示されたシステムのプロセスに供給されるインストラクションのセットは非常に限られている。単に文字列データと実数を現す数値データとが供給されるに過ぎない。

【0031】Wolfson 等によって開示されたシステムと同様の別のシステムは、C. Tschritzis et al., "KNOS: Knowledge Acquisition, Dissemination, and Manipulation Objects", ACM Trans. on office Information Systems, vol. 5, no. 1, pp. 96-112 (1987) に開示されている。Wolfson 等に記載されたこのシステムは、オブジェクト指向プログラミングの一般性を、Wolfson 等によって記述されたシステムに付加したものである。

【0032】オブジェクト指向プログラミングは、データ及びインストラクションをオブジェクトに組織化し、ここでデータはオブジェクトの状態を現し、インストラクションはそのオブジェクトが遂行しようとするタスク又はオペレーションに分類される。オブジェクト指向プログラミングは、コンピュータによって解決された問題がより容易に一連のコンピュータインストラクションに帰着さ

れうるような非常に有用な概念的なフレームワークを現している。

【0033】オブジェクト指向環境において作りだされたオブジェクトはクラスに分類される。ある1つのクラスのオブジェクトは同一の構造の状態を持ち、同一のオペレーションを遂行する。Tschritzis等によって開示されたシステムは、クラスの定義の可動性を与えないので、Tschritzis等によって記載された環境内において作りだされたある特定のクラスのオブジェクトは、その特定のクラスがそれに対し定義されたコンピュータシステムに向かって移動しようにすぎない。又Tschritzis等によって記述された環境内のオブジェクトは、それらのオブジェクトがその内部において移動しようとするネットワークのコンピュータシステムが同質であることを要求する形態において表されている。

【0034】Tschritzis等によって開示されたシステムは、更に、第1のプロセスすなわちヘッドプロセスがそれ自身のコピーをつくりだすことができるためのインストラクションを更に記述している。これらのコピーはリム(手足)と呼ばれる。Tschritzis等は、ヘッドプロセスの指令のもとにリムが遠隔のコンピュータシステムに送られることを教示する。しかしこれらのリムは活性ではない。リムについてのコンピュータのインストラクションは、ヘッドプロセスの指示においてのみ実行される。遠隔のコンピュータシステムに位置されたリムのプロセスの活動性の指示には、ヘッドプロセスがネットワーク通信メディアを横断してリムに指示を送出することを要求するので、大きなネットワークの場合、時間及びコストの浪費が大きくなる。

【0035】Tschritzis等によってキョウジュされたシステムにおいての別の非効率性は、小型のコンピュータ例えばパーソナルコンピュータが大型のコンピュータ例えば主フレームコンピュータを経て大きなネットワークに接続されている場合に見られる。一例として、ネットワークのいろいろなコンピュータに多くのリムを送出する場合、主コンピュータは、小型のコンピュータと大型のコンピュータとの間に多くの同一のリムを送らなければならない。ヘッドプロセスのコピーであるリムは、非常に大きくなることもあり、このようなシステムにおいての非効率性は非常に大きくなることもある。

【0036】Tschritzis等は、第2のプロセスに第1のプロセスへの参照をあたえることによって第1のプロセスが第2のプロセスと通信するシステムを開示している。参照は、あるオブジェクトを特定化しその特定化されたオブジェクトへのアクセスを許容するデータである。第2のプロセスは第1のプロセスへの参照を含んでいるので、第2のプロセスは、第1のプロセスに含まれる特定のコンピュータインストラクションを第1のプロセスが実行することを要求することができる。しかし、第1のプロセスが第2のプロセスへの参照を同時に取得



することなしに第1のプロセスへの参照を第2のプロセスに与えるので、第2のプロセスは第1のプロセスへの参照を持ち、第1のプロセスは第2のプロセスへの相互的なアクセスを持たない。このようなプロセスは悪意のあるプロセス、すなわち悪意のあるプログラムによって設計されたプロセス、あるいは、他のプロセスを損傷させ、相互的なアクセスを許容することなく第2のプロセスへのアクセスを行うように、不注意に設計されたシステムを許容する。

【0037】

【発明が解決しようとする課題】したがって、オブジェクト及びプロセスの新しいクラスを作りだし、それらの新しいクラスのためのクラス定義を含まないネットワーク内のコンピュータシステムに転送し処理するようなシステムは、Wolfson 等及び Tsichritzis等による業績にかかわらず、どちらにも開示されていないことになる。更に、Wolfson 等も、Tsichritzis等も、複雑で階層的なセキュリティシステムを具体化するメカニズムは開示していない。更に、Wolfson 等も、Tsichritzis等も、互いに相互作用は行うことができるが、相互的なアクセスを許容することなしに他のプロセスへのアクセスを得ることができるコンピュータシステムは開示しうることができない。

【0038】又、あるプロセスのいくつかの事実的なクローンをつくりだしてそれらのクローンをそれぞれのコンピュータシステムに送り込むことによっていくつかのコンピュータシステムに同時に移行することのできるコンピュータプロセスは、Wolfson 等にも Tsichritzis等にも開示されていない。Tsichritzis等によって教示されたリムは、コンピュータプロセスが遠隔コンピュータシステムにおいてリムの動作を制御するので、自律的ではない。更に、Wolfson 等にも Tsichritzis等にも、コンピュータ通信メディアを横断する冗長な情報の転送を最小にするそれぞれのコンピュータシステムへのクローンの効率的な転送を開示していない。

【0039】

【課題を解決するための手段】本発明は、エージェントクラス及びプレイスクラスを含む複数のオブジェクト指向クラスを定義し、前記オブジェクト指向クラス、該オブジェクト指向クラスのサブクラス及び go オペレーションを含む、コンピュータプロセスのためのインストラクションを形成し、該コンピュータネットワーク中のプロセッサによって前記インストラクションをインタープリートし、前記 go オペレーションに応答して、エージェントプロセスがプレイスプロセスに転送され、前記エージェントプロセスが、前記エージェントクラスの1つのメンバであり、前記プレイスプロセスは前記プレイスクラスの1つのメンバであることを特徴とする。

【0040】本発明は、前記エージェントプロセスにおいて前記 go オペレーションを形成することを特徴とす

る。

【0041】本発明は、前記プレイスのサブプレイスを形成し、前記サブプレイスは前記プレイスクラスの1つのメンバであり、前記プレイスは前記サブプレイスの1つのスーパープレイスであることを特徴とする。

【0042】本発明は、前記エージェントプロセスをオブジェクトのオーナーとして指定することを特徴とする。

【0043】本発明は、前記オブジェクト中にダイジェストを形成することを特徴とする。

【0044】本発明は、前記オブジェクトの代わりに前記プレイスプロセスにおいて第2のオブジェクトを相互変換し、前記第2のオブジェクトは、前記ダイジェストと同等の第2のダイジェストを有し、前記第1のオブジェクトは前記プレイスプロセスに転送されないことを特徴とする。

【0045】本発明は、前記プレイスプロセスをチケット手段によって指定することを特徴とする。

【0046】本発明は、前記チケット手段を前記エージェントプロセス内において形成することを特徴とする。

【0047】本発明は、前記チケット手段中において前記プレイスプロセスのアドレスを形成することを特徴とする。

【0048】本発明は、前記アドレスがテレアドレスであることを特徴とする。

【0049】本発明は、前記チケット手段中において前記プレイスプロセスのネームを形成することを特徴とする。

【0050】本発明は、前記ネームがテレネームであることを特徴とする。

【0051】本発明は、クラスオブジェクトのクラスを規定し、クラスオブジェクトを形成し、前記クラスオブジェクトは、(i) クラスオブジェクトの前記クラスの1つのメンバであり、(ii) 前記オブジェクト指向クラスの選択された1つのもののサブクラスである第1のクラスを規定し、前記第1のクラスの1つのメンバであるオブジェクトを形成し、前記エージェントプロセスは、前記オブジェクトを所有し、前記エージェントプロセスを前記プレイスプロセスに転送することが、前記オブジェクト及び前記クラスオブジェクトを前記プレイスプロセスに転送することを含むことを特徴とする。

【0052】本発明は、エージェントクラスとプレイスクラスとを含む複数のオブジェクト指向クラスを定義し、前記オブジェクト指向クラス、前記オブジェクト指向クラスのサブクラス及び send オペレーションを含むコンピュータプロセスのためのインストラクションを形成し、前記コンピュータのネットワーク中のプロセッサにおいて前記インストラクションをインタープリートする際に、前記 send オペレーションに応答してエージェントプロセスの1つのクローンが1つのプレイスプロセ



スに転送され、前記クローン及び前記エージェントプロセスは、各々前記エージェントクラスの1つのメンバであり、前記ブレイスプロセスは前記ブレイスクラスの1つのメンバであることを特徴とする。

【0053】本発明は、前記 send オペレーションを前記エージェントプロセス中において形成することを特徴とする。

【0054】本発明は、前記ブレイスのサブブレイスを形成し、前記サブブレイスは前記ブレイスクラスの1つのメンバであり、前記ブレイスは前記サブブレイスの1つのスーパーブレイスであることを特徴とする。

【0055】本発明は、前記エージェントプロセスをオブジェクトのオーナーとして指定することを特徴とする。

【0056】本発明は、前記クロノンの転送が前記プロジェクトのコピーを前記ブレイスプロセスに転送することを特徴とする。

【0057】本発明は、前記オブジェクト中にダイジェストを形成することを特徴とする。

【0058】本発明は、前記オブジェクトと異なる第2のオブジェクトを前記ブレイスプロセスにおいて前記コピーと相互変換し、前記ブレイスプロセスにおいて前記第2のオブジェクトは、前記ダイジェストと同等の第2のダイジェストを持ち、前記コピーは前記ブレイスプロセスに転送されないことを特徴とする。

【0059】本発明は、前記ブレイスプロセスをチケット手段によって指定することを特徴とする。

【0060】本発明は、前記エージェントプロセス中において前記チケット手段を形成することを特徴とする。

【0061】本発明は、前記ブレイスプロセスのアドレスを前記チケット手段中において形成することを特徴とする。

【0062】本発明は、前記アドレスがテレアドレスであることを特徴とする。

【0063】本発明は、前記チケット手段中において前記ブレイスプロセスの1つのネームを形成することを特徴とする。

【0064】本発明は、前記ネームがテレネームであることを特徴とする。

【0065】本発明は、前記 send オペレーションにตอบสนองして前記エージェントプロセスの第2のクロノンを第2のブレイスプロセスに転送し、前記第2のクロノンは、前記クロノンとは異なり、前記エージェントクラスの1つのメンバであり、前記第2のブレイスプロセスは前記ブレイスプロセスの1つのメンバであることを特徴とする。

【0066】本発明は、前記第1のクロノンを前記第1のブレイスプロセスに転送し、前記第2のクロノンを前記第2のブレイスプロセスに転送するためには、前記コンピュータネットワークの複数のコンピュータの内の単

一のコンピュータに前記第1及び第2のクロノンを転送することを必要であることを定め、前記第1のクロノンを前記単一のコンピュータに転送し、前記単一のコンピュータにおいて前記第1のクロノンから前記第2のクロノンを形成することを特徴とする。

【0067】本発明は、前記単一のコンピュータから前記ブレイスプロセスに前記第2のクロノンを転送し、前記単一のコンピュータから前記第2のブレイスプロセスに前記第2のクロノンを転送することを特徴とする。

【0068】本発明は、エージェントクラスを含む複数のオブジェクト指向クラスを定義し、前記オブジェクト指向クラス、前記オブジェクト指向クラスのサブクラス及びmeet オペレーションを含むコンピュータプロセスのためのインストラクションを形成し、前記コンピュータネットワーク中の1つのプロセッサにおいて前記インストラクションをインタープリットし、ミーティングブレイスプロセスが前記meet オペレーションにตอบสนองして、第2のエージェントプロセスへの第1のエージェントプロセスのアクセスを供与し、前記第1のエージェントプロセスへの前記第2のエージェントプロセスのアクセスも供与し、さらに、前記第1及び第2のエージェントプロセスが前記エージェントクラスのメンバであることを特徴とする。

【0069】本発明は、前記ミーティングブレイスプロセスが前記複数のオブジェクト指向クラス中のブレイスクラスの1つのメンバであり、前記第1及び第2のエージェントプロセスが前記ミーティングブレイスプロセスを占有することを特徴とする。

【0070】本発明は、第2のブレイスプロセスを形成し、第2のブレイスプロセスは前記ブレイスクラスの1つのメンバであり、前記第2のブレイスプロセスは前記ミーティングブレイスプロセスの1つのサブブレイスであり、前記ミーティングブレイスは前記第2のブレイスのスーパーブレイスであることを特徴とする。

【0071】本発明は、エージェントクラス、ブレイスクラス及びチケットクラスを含む複数のオブジェクト指向クラスを定義し、各々が前記ブレイスクラスの1つのメンバである、前記コンピュータネットワーク中の複数のブレイスプロセスを形成し、前記エージェントクラスの1つのメンバであって前記複数のブレイスプロセス中の第1のブレイスプロセスを占有する、エージェントプロセスを形成し、前記チケットクラスの1つのメンバであり、前記複数のブレイスプロセス中の前記第1のブレイスプロセスから第2のブレイスプロセスへの前記エージェントプロセスの移動を含むトリップを定義するチケットを形成することを特徴とする。

【0072】本発明は、ブレイスプロセスとパーミットクラスとを含む複数のオブジェクト指向クラスを定義し、前記ブレイスクラスの1つのメンバであるプロセスを形成し、前記パーミットクラスの1つのメンバであっ

て前記プロセスの1以上の能力を指定するパーミットを形成することを特徴とする。

【0073】本発明は、前記プロセスクラス中に go オペレーションを定義し、前記プロセスが前記 go オペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする。

【0074】本発明は、前記プロセスクラス中に send オペレーションを定義し、前記プロセスが前記 send オペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする。

【0075】本発明は、前記プロセスクラス中において charge オペレーションを定義し、前記プロセスが前記 charge オペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする。

【0076】本発明は、前記プロセスクラス中に terminate オペレーションを定義し、前記プロセスが前記 terminate オペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする。

【0077】本発明は、前記プロセスが前記プロセスと異なった第2のプロセスを作り出すことができるかを前記パーミット中において指定し、該第2のプロセスは前記プロセスクラスの1つのメンバであることを特徴とする。

【0078】本発明は、オブジェクト指向プレイスクラスを定義し、前記パーミット中に前記プロセスが前記プレイスプロセスの複数のメンバを作り出すことができるかを前記パーミット中において指定することを特徴とする。

【0079】本発明は、前記プロセスが失敗した時に前記プロセスが再スタートされるかを前記パーミット中において指定することを特徴とする。

【0080】本発明は、前記プロセスに割り当てられた処理量を前記パーミット中において指定することを特徴とする。

【0081】本発明は、前記プロセスが終了する時間を前記パーミット中において指定することを特徴とする。

【0082】本発明は、前記プロセスクラス中において制限オペレーションを定義し、前記クラス、前記クラスのサブクラス及び前記制限オペレーションを含むコンピュータプロセスのためのインストラクションを形成し、前記コンピュータネットワーク中のプロセッサにおいて前記インストラクションをインタープリットし、前記パーミットと異なる第2のパーミットが前記制限オペレーションにตอบสนองして形成され、前記第2のパーミットは前記パーミットクラスの1つのメンバであり、前記プロセスの前記1以上の能力の1つのグループを指定し、前記第2のパーミットによって指定された1以上の能力の前記グループに前記プロセスを制限することを特徴とする。

【0083】本発明は、コンピュータにおいて、クラス

(複数)の1つのクラスとサイテーション(複数)の1つのクラスを含む、複数のオブジェクト指向クラスを定義し、前記複数のクラスの前記クラスのメンバである1以上のクラスオブジェクトを形成し、前記第1のクラスオブジェクトを指定するとともに、前記クラスオブジェクトの内のどれが前記第1のオブジェクトに対して後方にコンパティブルであるかを指定するサイテーションを前記クラスオブジェクト内の第1のものの中に形成することを含む、インストラクションセットの種々のバージョンのプロセス(複数)をインタープリットする。

【0084】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、各々がゼロ又はそれ以上のエージェントプロセスのための前記コンピュータ内の1つのもののある場所である複数のプレイスプロセスを前記コンピュータネットワーク中に用意し、前記複数のプレイスプロセス中の目的点プレイスプロセスへのエージェントプロセスのトリップをチケット手段によって指定し、前記エージェントプロセス中の send オペレーションにตอบสนองして前記エージェントプロセスの1つのクローンを前記目的点プレイスプロセスに転送することを特徴とする。

【0085】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記複数のプレイスプロセス中の第2の目的点プレイスプロセスへの前記第2のエージェントプロセスの前記トリップと異なる第2のトリップを第2のチケット手段によって指定することを特徴とする。

【0086】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記第1のトリップを行なう前記クローンと前記第2のトリップを行なう第2のクローンとがどちらも前記コンピュータの内の単一のコンピュータに転送されるべきことを定め、前記第1のクローンを前記単一のコンピュータに移動させ、前記単一のコンピュータ中に、前記第1のクローンと異なる前記第2のクローンを形成することを特徴とする。

【0087】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記第1のクローンを前記目的点プレイスプロセスに転送することを特徴とする。

【0088】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記第2のクローンを前記第2の目的点プレイスプロセスに転送することを特徴とする。

【0089】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェントプロセス中の go オペレーションにตอบสนองして、前記チケット手段によって指定される前記目的点プレイスプロセスに前記エージェントプロセスを転送することを特徴とする。

【0090】本発明は、複数のコンピュータを有するコ

ンピュータネットワークにおいて、前記チケット手段中のネームによって前記目的点ブレイスプロセスを指定することを特徴とする。

【0091】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記ネームがテレネームであることを特徴とする。

【0092】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段中のアドレスによって前記目的点ブレイスプロセスを指定することを特徴とする。

【0093】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記アドレスがテレアドレスであることを特徴とする。

【0094】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスがその1つのメンバであるクラスのサイテーションによって前記目的点ブレイスプロセスを前記チケット手段中において指定することを特徴とする。

【0095】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスのアドレス、前記目的点ブレイスプロセスのネーム及び前記目的点ブレイスプロセスがその1つのメンバであるクラスのサイテーションの任意の組み合わせによって前記目的点ブレイスプロセスを前記チケット手段中において指定することを特徴とする。

【0096】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記トリップの最大の期間を前記チケット手段中に指定することを特徴とする。

【0097】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記トリップのための望ましい期間を前記チケット手段中に指定することを特徴とする。

【0098】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、転送手段を前記チケット手段中に指定し、該転送手段は前記クローンを転送するためのコンピュータ間通信手段の形式を指定することを特徴とする。

【0099】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記エージェントプロセスのデッドラインを、前記チケット手段中に含まれる目的点パーミットによって制御することを特徴とする。

【0100】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記エージェントプロセスが遂行することが許可されたオペレーションを、前記チケット手段に含まれる目的点パーミットによって制御することを特徴とする。

【0101】本発明は、複数のコンピュータを有するコ

ンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記エージェントプロセスが消費することが許可されたリソースを、前記チケット手段に含まれる目的点パーミットによって制御することを特徴とする。

【0102】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいての他のプロセスに対する前記目的点ブレイスプロセスにおいての前記エージェントプロセスの相対的なプライオリティを、前記チケット手段に含まれる目的点パーミットによって指定することを特徴とする。

【0103】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、あるプロセスが持つことを許可される能力をパーミット手段を介して制御することを特徴とする。

【0104】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段が固有のパーミットであることを特徴とする。

【0105】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段がローカルパーミットであることを特徴とする。

【0106】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段が一時的なパーミットであることを特徴とする。

【0107】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が前記プロセスの消費しうるリソースからなることを特徴とする。

【0108】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、その後ではプロセスが進行しえないデッドラインを含むことを特徴とする。

【0109】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が他のプロセスに対する前記プロセスの相対的なプライオリティを含むことを特徴とする。

【0110】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、前記プロセスが選択されたオペレーションを遂行する許可を含むことを特徴とする。

【0111】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェントプロセスがオブジェクトを所有することを特徴とする。

【0112】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記オブジェクトがダイジェストを有することを特徴とする。

【0113】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェントプロセスのクローンを転送する前記工程が前記オブジェクトのコピーを転送することを含むことを特徴とする。

【0114】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記コピーを第2のオブジェクトに相互変換し、該第2のオブジェクトは、最初に述べた前記ダイジェストと同等のダイジェストを有し、前記コピーは前記目的点ブレイスプロセスには転送されないことを特徴とする。

【0115】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記オブジェクトの代わりに、相互変換されたオブジェクトを使用して、前記オブジェクトを前記目的点ブレイスプロセスに転送する必要を除くことを特徴とする。

【0116】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記転送工程が前記目的点ブレイスプロセスにエンタする工程を含むことを特徴とする。

【0117】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記転送ステップが前記ソースブレイスプロセスをエクシットする工程を含むことを特徴とする。

【0118】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスがミーティングブレイスプロセスであり、前記ミーティングブレイスはリクエストエージェントプロセスとペティショントエージェントプロセスとの間のミーティングを取り計らうことを特徴とする。

【0119】本発明は、コンピュータにおいて、第1のエージェントプロセスと第2のエージェントプロセスとを用意し、ペティション手段によって、前記第1のエージェントプロセスと第2のエージェントプロセスとの間のミーティングを指定し、前記ペティション手段によって定義された前記第1のエージェントプロセスと第2のエージェントプロセスとの間の前記ミーティングを取り計らうことを特徴とする。

【0120】本発明は、コンピュータにおいて、前記第1のエージェントプロセス中に前記ペティション手段を形成することを特徴とする。

【0121】本発明は、コンピュータにおいて、ミーティングを特定する前記工程は、前記ペティション手段中において前記第2のエージェントプロセスを指定することを特徴とする。

【0122】本発明は、コンピュータにおいて、前記第2のエージェントプロセスを指定する前記工程が、前記第2のエージェントをネームによって前記ペティション手段中において指定することを特徴とする。

【0123】本発明は、コンピュータにおいて、前記ネームがテレネームであることを特徴とする。

【0124】本発明は、コンピュータにおいて、前記第2のエージェントプロセスを指定する前記工程が、前記

第2のエージェントプロセスがその1つのメンバであるクラスを前記ペティション手段中において指定することを特徴とする。

【0125】本発明は、コンピュータにおいて、前記第2のエージェントプロセスを指定する前記工程は、前記第2のエージェントプロセスがその1つのメンバであるクラスのサイテーションを前記ペティション手段中において指定することを特徴とする。

【0126】本発明は、コンピュータにおいて、ミーティングを指定する前記工程が、前記ミーティングを取り計らうための最大の期間を前記ペティション手段中において指定することを特徴とする。

【0127】本発明は、コンピュータにおいて、前記ミーティングを取り計らう前記工程は、前記第2のエージェントプロセスへのリファレンスを前記第1のエージェントプロセスに供与することを特徴とする。

【0128】本発明は、コンピュータにおいて、前記ミーティングを取り計らう前記工程が、前記第1のエージェントプロセスへのリファレンスを前記第2のエージェントプロセスに対して供与することを特徴とする。

【0129】本発明は、コンピュータにおいて、前記第2のエージェントプロセス中のオペレーションの遂行を前記第1のエージェントプロセス中においてインストラクションに応答して生じさせることを特徴とする。

【0130】本発明は、コンピュータにおいて、前記第1のエージェントプロセスがオブジェクトを所有することを特徴とする。

【0131】本発明は、コンピュータにおいて、前記オブジェクトへのリファレンスを前記第2のエージェントプロセスに対して供与することを特徴とする。

【0132】本発明は、コンピュータにおいて、前記リファレンスが保護されるリファレンスであることを特徴とする。

【0133】本発明は、コンピュータにおいて、前記オブジェクトのコピーへのリファレンスを前記第2のエージェントプロセスに供与することを特徴とする。

【0134】本発明は、コンピュータにおいて、前記第2のエージェントプロセスがオブジェクトを所有することを特徴とする。

【0135】本発明は、コンピュータにおいて、前記オブジェクトへのリファレンスを前記第1のエージェントプロセスに対して供与することを特徴とする。

【0136】本発明は、コンピュータにおいて、前記リファレンスがプロテクトされたリファレンスであることを特徴とする。

【0137】本発明は、コンピュータにおいて、前記オブジェクトのコピーへのリファレンスを前記第1のエージェントプロセスに対して供与することを特徴とする。

【0138】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、チケット手段及び s

end オペレーションを有するエージェント手段と、前記複数のコンピュータの1つにおいてその各々が動作する複数のブレイス手段とを有し、前記エージェント手段は前記複数のブレイス手段中の第1のブレイス手段であり、前記チケット手段は、前記複数のブレイス手段中の目的点ブレイス手段への前記エージェント手段のトリップを指定し、前記 send オペレーションは、前記エージェント手段の1つのクローンを前記目的点手段に転送することを特徴とする。

【0139】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段が、前記エージェント手段のための第2の目的点ブレイス手段への第2のトリップを指定する、前記チケット手段と異なる第2のチケット手段を、含むことを特徴とする。

【0140】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記 send オペレーションの遂行によって前記エージェント手段の第2のクローンが前記第2の目的点ブレイス手段に転送されることを特徴とする。

【0141】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記トリップ及び前記第2のトリップが、前記複数のコンピュータ中のある単一のコンピュータへの転送を含み、前記クローンを前記目的点ブレイス手段に転送し、前記第2のクローンを前記第2の目的点手段に転送する際に、前記 send オペレーションの遂行によって、(a) 前記第1のクローンが前記単一のコンピュータに転送され、(b) 前記単一のコンピュータ中において前記第1のクローンから前記第2のクローンが形成されることを特徴とする。

【0142】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段が go オペレーションを含み、前記 go オペレーションは、第3のチケット手段によって指定された第3の目的点ブレイス手段に前記エージェント手段を転送することを特徴とする。

【0143】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記目的点ブレイス手段を特定するネームを含むことを特徴とする。

【0144】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記ネームがテレネームであることを特徴とする。

【0145】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記目的点ブレイス手段のためのアドレスを含むことを特徴とする。

【0146】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記アドレスがテレアドレスであることを特徴とする。

【0147】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記目的点ブレイス手段をその1つのメンバとするクラスを指定するサイテーション手段を含むことを特徴とする。

【0148】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が、前記目的点ブレイス手段のアドレスと、前記目的点ブレイス手段のネームと、前記目的点ブレイス手段をその1つのメンバとするクラスを指定するサイテーション手段とからなる群中より選ばれた任意の組合せを含むことを特徴とする。

【0149】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記トリップのための最大の期間を指定する手段を含むことを特徴とする。

【0150】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記トリップのための望ましい期間を指定する手段を含むことを特徴とする。

【0151】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が、前記トリップ手段を完成させるコンピュータ間通信手段の形式を指定するウェイ手段を含むことを特徴とする。

【0152】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が、前記目的点ブレイス手段においての前記エージェント手段のためのデッドラインを制御するパーミット手段を含むことを特徴とする。

【0153】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段が前記目的点ブレイス手段において遂行することが許可されているオペレーションを制御するためのパーミット手段を前記チケット手段を有することを特徴とする。

【0154】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイス手段において前記エージェント手段が消費することが許可されているリソースを制御するためのパーミット手段を前記チケット手段が有することを特徴とする。

【0155】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイス手段においての他のエージェント手段に対する前記目的点ブレイス手段においての前記エージェント手段の相対的なプライオリティを指定するためのパーミット手段を前記チケット手段が有することを特徴とする。

【0156】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、エージェント手段が持つことが許可される能力を制御するためのパーミット手段を有することを特徴とする。

【0157】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段が固有のパーミットであることを特徴とする。

【0158】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段がローカルパーミットであることを特徴とする。

【0159】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段が一時的なパーミットであることを特徴とする。

【0160】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が前記エージェント手段の消費するリソースを含むことを特徴とする。

【0161】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、前記エージェント手段がそれから後進行することのできないデッドラインを含むことを特徴とする。

【0162】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、他のエージェント手段を含み、前記能力は、前記他のエージェント手段に対する前記エージェント手段の相対的なプライオリティを含むことを特徴とする。

【0163】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、前記エージェント手段が選択されたオペレーションを遂行する許可を含むことを特徴とする。

【0164】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段がオブジェクトを所有することを特徴とする。

【0165】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記オブジェクトがダイジェストを有することを特徴とする。

【0166】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記オブジェクトのコピーが前記クローンとともに、前記 send オペレーションの遂行によって転送されることを特徴とする。

【0167】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイス手段において第2のオブジェクトは前記コピーに相互変換され、前記第2のオブジェクトは、前記ダイジェストと同等のダイジェストを持つことによって前記コピーを前記目的点ブレイス手段に転送することを不要とすることを特徴とする。

【0168】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記 send オペレーションが前記目的点ブレイス手段においての前記コピーの代わりに相互変換されたオブジェクトを使用することによって、前記オブジェクトの前記コピーを前記目的点ブレイス手段に転送する必要を除くことを特徴とする。

【0169】本発明は、複数のコンピュータを有するコ

ンピュータネットワークにおいて、前記ブレイス手段は前記ブレイス手段にエンタするための手段を有することを特徴とする。

【0170】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記ブレイス手段が前記ブレイス手段をエグジットする手段を有することを特徴とする。

【0171】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記ブレイス手段がミーティングブレイス手段を含み、前記ミーティングブレイス手段はリクエストエージェント手段とレスポンドエージェント手段との間のミーティングを取り計らうことを特徴とする。

【0172】本発明は、コンピュータにおいて、 meet オペレーションを有するミーティングブレイス手段を有し、該ミーティングブレイスは、複数のエージェント手段のための場所であり、更に、ベティッション手段と、前記複数のエージェント手段の内の第1のエージェント手段と、前記複数のエージェント手段の中の第2のエージェント手段とを有し、前記第1のエージェント手段及び第2のエージェント手段は前記複数のエージェント手段の内のどれともミートでき、前記ベティッション手段は前記第1のエージェント手段と第2のエージェント手段との間のミーティングを指定し、前記ミーティング手段は前記 meet オペレーションに応答して前記ミーティングを取り計らうことを特徴とする。

【0173】本発明は、コンピュータにおいて、前記第1のエージェント手段が前記ベティッション手段を有することを特徴とする。

【0174】本発明は、コンピュータにおいて、前記ベティッション手段が前記第2のエージェントプロセスを規定することによって前記ミーティングを指定することを特徴とする。

【0175】本発明は、コンピュータにおいて、前記ベティッション手段が前記第2のエージェント手段を指定するネームを含むことを特徴とする。

【0176】本発明は、コンピュータにおいて、前記ネームがテレネームであることを特徴とする。

【0177】本発明は、コンピュータにおいて、前記第2のエージェントプロセスがその1つのメンバであるクラスを指定することによって前記ベティッション手段が前記第2のエージェントを指定することを特徴とする。

【0178】本発明は、コンピュータにおいて、前記ベティッション手段がサイテーションを含み、該サイテーションは、前記第2のエージェントプロセスがその1つのメンバであるクラスを指定することを特徴とする。

【0179】本発明は、コンピュータにおいて、前記ベティッション手段が前記ミーティングを取り計らうための最大の期間を規定することを特徴とする。

【0180】本発明は、コンピュータにおいて、前記 m

eeet オペレーションが前記第2のエージェント手段へのリファレンスを前記第1のエージェント手段に供与することを特徴とする。

【0181】本発明は、コンピュータにおいて、前記 meeeet オペレーションが前記第1のエージェント手段へのリファレンスを前記第2のエージェント手段に供与することを特徴とする。

【0182】本発明は、コンピュータにおいて、前記第1のエージェント手段が前記第2のエージェント手段中のオペレーションの遂行を生じさせることを特徴とする。

【0183】本発明は、コンピュータにおいて、前記第1のエージェント手段がオブジェクトを所有することを特徴とする。

【0184】本発明は、コンピュータにおいて、前記第1のエージェント手段が前記オブジェクトへのリファレンスを前記第2のエージェント手段に供与することを特徴とする。

【0185】本発明は、コンピュータにおいて、前記リファレンスがプロテクトされたリファレンスであることを特徴とする。

【0186】本発明は、コンピュータにおいて、前記第1のエージェント手段が前記オブジェクトのコピーへのリファレンスを前記第2のエージェント手段に供与することを特徴とする。

【0187】本発明は、コンピュータにおいて、前記第2のエージェント手段がオブジェクトを所有することを特徴とする。

【0188】本発明は、コンピュータにおいて、前記第2のエージェント手段が前記オブジェクトへのリファレンスを前記第1のエージェント手段に供与することを特徴とする。

【0189】本発明は、コンピュータにおいて、前記リファレンスがプロテクトされたリファレンスであることを特徴とする。

【0190】本発明は、コンピュータにおいて、前記第2のエージェント手段が前記オブジェクトのコピーへのリファレンスを前記第1のエージェント手段に供与することを特徴とする。

【0191】本発明は、1以上のコンピュータを有するコンピュータネットワークにおいて、第1のエンジンプロセスから第2のエンジンプロセスにデータを移送するデータ移送方法において、(a)前記第1のエンジンプロセス及び第2のエンジンプロセスを含む1以上のエージェントプロセスを実行するための実行手段を用意し、各々の前記エージェントプロセスは、コンピュータインストラクションセットからの複数のインストラクションを含み、実行状態を持つものとし、(b)前記コンピュータインストラクションセット中に go インストラクションを用意し、該 go インストラクションは、前記第1

のエンジンプロセス中において実行される第1のエージェントプロセスに含まれるものとし、前記 go インストラクションの遂行によって、(i)前記第1のエンジンプロセスによる前記第1のエージェントプロセスの実行の中断と、(ii)前記第1のエージェントプロセスの前記実行状態が保存されるような前記第1のエージェントプロセスの表現と、(iii)前記第1のエンジンプロセスから第2のエンジンプロセスへの前記第1のエージェントプロセスの前記表現の移送と、(iv)前記第2のエンジンプロセスによる前記第1のエージェントプロセスの実行の再開とを生じさせ、(c)前記第1のエージェントプロセスによる前記エージェントプロセスの実行を生じさせることによって前記 go インストラクションの遂行を生じさせることを特徴とする。

【0192】本発明は、前記コンピュータインストラクションセットがオブジェクト指向であることを特徴とする。

【0193】本発明は、前記複数のエージェントプロセスが前記オブジェクト指向コンピュータインストラクションセットのオブジェクトであることを特徴とする。

【0194】本発明は、前記 go インストラクションの遂行を生じさせる前に前記第1のエージェントプロセスにデータを付加し、前記第1のエージェントプロセスの前記表現が前記データを含み、前記第2のエンジンプロセスへの前記第1のエージェントプロセスの前記表現の移送が前記データを移送を含むことを特徴とする。

【0195】本発明は、前記第1のエージェントプロセスが、前記第1のエンジンプロセスから前記第2のエンジンプロセスに転送されるべきメッセージを表すデータを含み、前記 go インストラクションの実行によって生ずる前記第1のエージェントプロセスの移送が前記第1のエンジンプロセスから前記第2のエンジンプロセスへの前記データの移送を生じさせることを特徴とする。

【0196】本発明は、前記第1のエンジンプロセスが、第1のコンピュータにおいて実行され、前記第2のエンジンプロセスが第2のコンピュータによって実行され、前記第1及び第2のコンピュータが前記コンピュータネットワークの一部であることを特徴とする。

【0197】本発明は、第1のエンジンプロセスから1以上のエンジンプロセスにデータを移送するデータ移送方法において、(a)コンピュータインストラクションセットからのインストラクションを含む複数のエージェントプロセスを実行するための手段を用意し、該エージェントプロセスを実行する手段が、前記第1のエンジンプロセス及び1以上のエンジンプロセスを含み、各々の該エージェントプロセスは実行状態を含み、(b)前記コンピュータインストラクションセット中に send インストラクションを用意し、該 send インストラクションは、前記複数のエージェントプロセス中の第1のエージェントプロセス中に含まれるようにし、前記 send インストラ



クションの実行は、(i) 前記第1のエージェントプロセスの1以上のコピーを形成し、これらのコピーが、前記第1のエージェントプロセスの前記実行状態を保存するとともにそれを含み、(i i) 前記第1のエージェントプロセスのコピーの各々を前記第1のエンジンプロセスから前記1以上のエンジンプロセスのそれぞれに移送し、(i i i) 前記第1のエージェントプロセスの前記コピーの各々を前記1以上のエンジンプロセスのそれぞれの1つによって実行して前記第1のエージェントプロセスの再開された実行をシミュレートすることを特徴とする。

【0198】本発明は、前記第1のエージェントプロセスの前記コピーの2以上が前記第1のエンジンプロセスから単一の第2のエンジンプロセスに移送されるべき条件のもとにおいて、前記第1のエージェントプロセスの前記2以上のコピーの移送が、前記第1のエージェントプロセスの単一のコピーを前記第2のエンジンプロセスに移送する工程と、前記第2のエンジンプロセス中において前記単一のコピーから、前記第1のエージェントプロセスの前記2以上のコピーを形成する工程とからなることを特徴とする。

【0199】本発明は、あるコンピュータシステムにおいて実行されている第1のエージェントプロセスから、該コンピュータシステムによって実行されている第2のエージェントプロセスにデータを転送する際に、前記第1のエージェントプロセス及び第2のエージェントプロセスがあるプレイスプロセスの占有者であるようにしてデータを転送するデータ転送方法において、前記第1のエージェントプロセスが送出するミートインストラクションに応答して、前記第2のエージェントプロセスによるプロシージャの実行を生じさせる工程を含み、前記プロシージャは、前記第2のエージェントプロセスの一部であり、前記第2のエージェントコンピュータプロセスに含まれるコンピュータインストラクションのコレクションを含み、前記第2のエージェントプロセスによって送出される、前記プロシージャの一部である第2のインストラクションに応答して、前記第2のエージェントプロセスへのアクセス手段を前記第1のエージェントプロセスに供与する工程を含むことを特徴とする。

【0200】本発明は、前記第2のエージェントプロセスによるプロシージャの実行を生じさせる前記工程は、前記第1のエージェントプロセスによって送出される前記ミートインストラクションに応答して前記プレイスプロセスによる第2のプロシージャの実行を生じさせることを含み、前記プレイスプロセスによる前記第2のプロシージャの実行が、前記プロシージャを前記第2のエージェントプロセスによって実行させるインストラクションを送出することを特徴とする。

【0201】本発明は、前記第2のエージェントプロセスへのアクセス手段が前記プレイスプロセスによって前

記第1のエージェントプロセスに供与されることを特徴とする。

【0202】本発明は、近似的に、前記プレイスプロセスが前記第2のエージェントプロセスへの前記アクセス手段を前記第1のエージェントプロセスに供与する時点において、前記プレイスプロセスが前記第1のエージェントプロセスへのアクセス手段を前記第2のエージェントプロセスに供与することを特徴とする。

【0203】本発明は、第1のCPU及び第1のメモリを含む第1のコンピュータシステムから、第2のCPU及び第2のメモリを含む第2のコンピュータシステムへ第1のコンピュータプロセスを移送するプロセスの移送方法において、ある実行状態を持つ前記第1のコンピュータプロセスの実行を前記第1のCPUにおいて開始し、前記第1のCPU内において前記第1のコンピュータプロセスの実行を中断し、前記第1のコンピュータプロセスをデータとして前記第1のメモリ中に表現し、前記データは前記第1のコンピュータプロセスの実行が中断された時点における前記第1のコンピュータプロセスの前記実行状態を含むものとし、前記データを前記第1のメモリから前記第2のメモリに移送し、前記データ中に表現された前記実行状態を有する第2のコンピュータプロセスを前記データから前記第2のコンピュータシステム上に形成し、前記第2のコンピュータプロセスを実行することによって前記第2のCPU内において前記第1のコンピュータシステムの実行の再開を実効的にシミュレートすることを特徴とする。

【0204】

【作用】本発明では、解釈されオブジェクト指向されたコンピュータインストラクションの一群が、分散型コンピュータシステムにおいて実行される新規なコンピュータプロセスを作りだすために用いられる。このコンピュータインストラクションの一群を利用する特別なプロセスは、分散型コンピュータシステム中において作動するエンジンによって活性にされる。エンジンは分散型コンピュータシステム内において特別なプロセスのインストラクションを解釈し遂行する。

【0205】分散型コンピュータシステム中の各々のエンジンは、あるプロセスを規定するインストラクションを一様に解釈する。換言すれば、あるプロセスを含み、第1のエンジンによって解釈される、インストラクションは、第1のエンジンがその内部において作動しているコンピュータシステムの特別な形態に依存しない。したがってこれらのインストラクションは、第2のエンジンによって、たとえ第1のエンジン及び第2のエンジンが2つの別々のコンピュータシステムにおいて作動しており、これらのコンピュータシステムのオペレーティングシステム及びハードウェアがその他の点では、一般に互いにコンパティビリティを持たないとしても、移動され解釈される。解釈されるインストラクションは、エンジ



ンがその内部において作動しているコンピュータシステムの通信、格納、計算及びその他のサブシステムへのインターフェースを用いて実現（インプリメント）される。

【0206】これらのインターフェースは、集散的に本発明の一実施例の一部を構成するものであり、この開示の一部とされて引用によって全体が本明細書に組み込まれるアペンディックスCに記載されている。

【0207】2つ以上のエンジンが1つのネットワークを構成するように相互接続される。ネットワークは本発明のコンピュータプロセス（複数）がその内部において移動する1つの宇宙である。一実施例によればネットワークは、他のネットワークのクライアントと通常考えられ、その一部分ではないとされるコンピュータシステム（複数）を網羅する。たとえば本発明の一実施例は、ネットワークによって接続されたワークステーション（複数）を網羅する。本明細書においてワークステーションという用語は、他のコンピュータシステムへの情報の転送と異なった目的のために使用されるコンピュータシステムを記述するために用いられている。本発明のエンジン（複数）は、コンピュータプロセス（複数）をその間において移動させることができるように相互接続されている。

【0208】ある特別なコンピュータプロセスを転送するために、そのコンピュータシステムは、中断され、コンピュータシステムの実行状態が保存される。コンピュータプロセスのインストラクション、保存された実行状態及びコンピュータプロセスによって所有されるオブジェクトは、パッケージされ又はエンコード（符号化）される、データのストリングを生成する。このデータのストリングは、1つのネットワークを形成するために用いられるすべての標準的な通信手段によって転送することができる。一実施例によればデータストリングは、この明細書の一部とされ、引用によって全体がこの明細書に組み込まれるアペンディックスDに記載されたプロトコルによって指定されるエンジンの間において転送される。

【0209】データストリングは、ネットワークの目的点となるコンピュータシステムに転送されると、デコードされ、目的点コンピュータシステム中において1つのコンピュータプロセスを生じさせる。デコードされたコンピュータプロセスは、前述したようにエンコードされたオブジェクトを含み、保存された実行状態を有している。目的点のコンピュータシステムは、コンピュータプロセスの実行を再開する。

【0210】コンピュータプロセスのインストラクションは、データオブジェクトの定義、生成及び操作と、目的点のコンピュータシステムによって実行される他のコンピュータシステムとの相互作用とを含む複雑なオペレーションを遂行するために、目的点のコンピュータシステムによって実行される。コンピュータプロセスはネッ

トワーク全体を通じて一様に解釈され、コンピュータシステムの間に転送されるようにエンコードされかつデコードされるので、本発明のコンピュータプロセス（複数）は、コンピュータ間の通信において新しいレベルの機能性及び多様性を供与する。

【0211】一実施例によれば、本発明のコンピュータインストラクションのセットに組み込まれた2つのクラスは、エージェントクラスとプレイスクラスである。エージェントクラスを用いて形成されたインストラクションは、エンジンによって解釈され、エージェントプロセス（本明細書中において単にエージェント）と呼ばれることもある。エージェントは、エージェントが生成されたときにエンジンがその実行を開始するという点でアクティブオブジェクトである。エージェントクラスは、エージェントが（i）それ自身を検査して変更し、（i i）ネットワーク中の以下にそれを詳細に説明する第1のプレイスプロセスから同じネットワーク中の第2のプレイスプロセスまでそれ自身を転送し、（i i i）第2のプレイスプロセスに見出される他のエージェントと相互作用することを可能にするインストラクション（複数）を供与する。第1のプレイスプロセス及び第2のプレイスプロセスはネットワークの2つの別々のコンピュータシステム内において実行される。したがってあるエージェントは第1のコンピュータシステムから第2のコンピュータシステムに移動することができる。

【0212】あるエージェントは、第1のプレイスプロセスから第2のプレイスプロセスにまでエージェントとともに運ばれる情報を有することができる。その他に、以下にその詳細に説明されるコンピュータインストラクションのセットの拡張性、不変性及び機能性によって、1つのエージェントを含むインストラクションにしたがって、そのエージェント及びそれによって運ばれる情報がどのようにそしてどの目的点に移動しているのかを定める上の膨大な融通性及び制御可能性が提供される。

【0213】エージェントの力はパーミットによってバランスされる。パーミットは、特別な場合に特別なエージェントの特別な能力を制限する。エージェントのパーミットは、そのエージェントクラスについて定義されたいくつかのオペレーションのうちどのオペレーションがエージェントによって遂行され得るかを指定する。さらにパーミットは、エージェントが消費し得る処理依存数及びエージェントが消滅する時間とを制限する。その上に、パーミットは、他のエージェントに対するエージェントの相対的な実行プライオリティを指定する。

【0214】プレイスクラスを用いて形成されたインストラクションは、エンジンによって解釈され、本明細書において単にプレイスとも呼ばれるプレイスプロセスを形成する。プレイスはまたアクティブオブジェクトでありプレイスクラスは、あるプレイスがそれ自身を検査して変更しエージェントのための場所とエージェントが相

相互作用する際のコンテキストとして役立ち得るようにするインスタクションを供与する。エージェントは各々のそれぞれ単一のプレイスを占有する。また各々のプレイスは単一の他のプレイスも占有することができる。

【0215】プレイスはエージェントのためのある度合いのプライバシー及びセキュリティを供与する。一例として、他のエージェントとのコンタクトをさけるように構成されたエージェントは、一般に他のエージェントには知られてないか又は他のエージェントの立ち入りを拒絶するあるプレイスを占有することができる。逆に、ある大きな数のエージェントに公にサービスを供与するように構成されたエージェントは、他のエージェントに広く知られていて他のエージェントの立ち入りを許容するプレイスを占有することができる。

【0216】エージェントはある距離において相互作用することはできない。すなわちいかなる2つのエージェントもそれらが同一のプレイスを占有するまでないかぎり相互作用できない。第1のプレイスを占有している第1のエージェントは、第2のプレイスを占有している第2のエージェントと相互作用するためには、オペレーション"go"に関連して以下に詳細に説明するように第1のエージェントを第2のプレイスに転送させるインスタクションが、第1のプレイスを占有している第1のエージェントによって送出される。第1及び第2のエージェントの両者が第2のプレイスを占有している間に、第1のエージェントは第2のエージェントと相互作用することができる。

【0217】したがって、本発明のプロセスは、リモートプログラミングの新規なインプリメンテーションであり、より知られているリモートプロシージャコーリングパラダイムのインプリメンテーションではない。リモートプログラミングは、(i) ネットワーク通信の媒体を横切る通信を必要とせずにプロセス(複数)が相互作用することを可能とし、(ii) ハイレイテンシ(high-latency)をもつことが多いネットワーク通信媒体を横切る通信を除くことによってプロセス間の相互作用の性能を改善することによって、リモートプロシージャコーリングを改良する。

【0218】第1のコンピュータシステム中の第1のプレイスプロセスから、第1のコンピュータシステム又は第2のコンピュータシステム中の第2のプレイスプロセスへのエージェントプロセスの移動はトリップと呼ばれる。エージェントは、エージェントクラス中において定義されたオペレーション"go"を用いることによってトリップを開始する。エージェントは、トリップを定義するチケットをオペレーション"go"に対するアーギュメントとして作り出し委託することによってネットワークを通る移動又はコンピュータシステム内の移動を制御する。一実施例によれば、チケットは、エージェントがそれにむかって移動するべきプレイス、エージェントが

それによって移動する"way"、トリップが完了する時間量及びトリップの緊急さの表示、すなわち同時にスケジュールされていることのある他のエージェントによる他のトリップに対するトリップの相対的なプライオリティの表示を指定している。

【0219】チケットは、エージェントがそれにむかって移動しているプレイスのアドレス、ネーム、クラス又はアドレス、ネーム、及びクラスの任意の組合せを指定することによって、目的点プレイスを特定することができる。トリップの目的点は、チケットによって許容された時間内のエージェントの立ち入りを許容する特定されたアドレス、ネーム及び/又はクラスの任意のプレイスである。

【0220】オペレーション"go"においてエージェントは第1のプレイスから第2のプレイスに移動するのは次のようにして行われる。(i) エージェントは中断され、第1のエージェントの実行状態が保存される。(ii) エージェントは標準形式ですなわちオクテットストリーム(エージェントの保存された実行状態の表現を含む)によって表される。(iii) 標準形式は第1のプレイスから第2のプレイスに転送される(これはポテンシャルには、第1のコンピュータから第2のコンピュータへの標準形式の移送を含む)。

【0221】(iv) 第2のプレイスにあるエージェントは、標準形式で表された実行状態を含めて標準形式から形成される。(v) 標準形式であらわされた実行状態を最初にもっているエージェントの解釈が再開される。

【0222】前述したように、本発明のコンピュータインスタクションのセットは、オブジェクト指向である。したがって本発明にしたがって形成されたすべてのオブジェクトはクラス(複数)に組織される。本発明のすべてのクラスは、エージェントとともに移動することのできるデータオブジェクトによって表される。したがって、あるプレイス内において定義されていないクラスも、単にエージェントがクラスを定義し、対応するクラスオブジェクトをそのプレイスに転送することによって、そのプレイスにむかって移動するエージェントによって使用することができる。

【0223】本発明の一実施例による各々のオブジェクトは、プロセスすなわちエージェント又はプレイスによって所有される。あるエージェントが第1のプレイスから第2のプレイスに移動する時に、そのエージェントによって所有されるすべてのオブジェクトは、実効的に、エージェントとともに第2のプレイスに移動する。しかし以下に説明するように、本発明の一実施例によれば、第2のプレイスをすでに占有しているオブジェクトと同等のオブジェクトは、転送されない。エージェントによって所有されるオブジェクトの他に、それらのオブジェクトがメンバであるクラスを定義するすべてのクラスオブジェクトは、エージェントとともに第2のプレイスに

移動される。クラスオブジェクトは、オブジェクトの1つのクラスを表す、以下及びアペンディックスAに説明されるコンピュータインストラクションのセットにしたがって構成されたオブジェクトである。

【0224】したがって、第2のプレイスに移動するエージェントによって所有されるオブジェクトが第2のプレイスにおいて定義されていないクラスのメンバであった場合にも、そのオブジェクトは、エージェントとともに移動することができる。

【0225】それは、エージェントが、オブジェクトを1つのメンバとするクラスを定義するクラスオブジェクトも第2のプレイスに運ぶからである。従来の技術によれば、第1のコンピュータから第2のコンピュータに移動するプロセスは、第2のコンピュータ上において定義されたクラスに所属するデータオブジェクトしか処理することができなかった。クラスの定義は通常は従来技術によれば移動するプロセスとともに転送されなかった。第1のプレイスにおいて新しいクラスを形成することができ、これらの新しいクラスのメンバは、これらの新しいクラスが定義されていない他のプレイスに自由に移動するので、本発明は、従来技術にはみられない拡張性、可動性及び不変性のレベルをエージェントに供与する。

【0226】本発明によれば、多数のオブジェクトクラスが多数のプレイスにおいて定義されるので、これらのプレイスに移動させる必要はない。本発明の一実施例によれば、クラスオブジェクト（複数）及び多くの場所において見出されることがあり得る他のオブジェクト（複数）は、相互変換可能とされている。

【0227】相互変換可能な（インターチェンジャブルな）オブジェクトは各々のダイジェストを有している。第1のプレイスから第2のプレイスに移動するエージェントによって所有された相互変換可能なオブジェクトは、エージェントとともに移動しない。そうではなく、相互変換可能なオブジェクトのダイジェストがエージェントとともに第2のプレイスに移動する。エージェントが第2のプレイスに到達すると、第2のプレイスを含むコンピュータシステムの相互変換可能なオブジェクト

（複数）は、これらの相互変換可能なオブジェクトのどれかがエージェントとともに転送されてきたダイジェストと等しいダイジェストをもつか否かを定めるために検査される。第2のプレイスを含むコンピュータシステムにおいてそのような相互変換可能なオブジェクトが見出された場合、その相互変換可能なオブジェクトは、第1のプレイスに残された相互変換可能なオブジェクトと取替えられる。

【0228】しかし、第2のプレイスにそのような相互変換可能なオブジェクトが見出されなかった場合、第1のプレイスに残されていた相互変換可能なオブジェクトが第2のプレイスに移動される。

【0229】これにより、同等なオブジェクトが目的点

のプレイスに存在する場合において、ネットワーク通信媒体を横切るオブジェクトの移動が避けられる。特に、クラスオブジェクトが相互変換可能なため、第2のプレイスに移動するエージェントが、第2のプレイスにおいて定義されていないクラスを定義するクラスオブジェクトのみの第2のプレイスへの移動を生じさせる。ネットワークを通るクラスオブジェクト（複数）の必要な運動を避けることによって、あるエージェントによって所有されたオブジェクトが所属する1以上のクラスの定義を含まないプレイスへのこれらのオブジェクトの移動が実用的にされる。

【0230】本発明の一実施例によればクラスはサイテーションによって特定される。異なったベンダ（供給者）によって供給される多くのコンピュータシステムを有するコンピュータネットワークにおいて、エージェントプロセスがそれにむかって、又はそれから移動することのできるいろいろのコンピュータシステム上に、ここに開示されたインストラクションセットの色々なバージョンをインプリメントすることができる。サイテーションは、相互に対し前方向又は後方方向にコンパティブルなクラス又はオブジェクトを特定するために用いられるオブジェクトである。したがって第1のクラスのオブジェクト又は第1のクラス自身の使用を必要とするあるインストラクションは、第2のクラスが第1のクラスに対して後方方向にコンパティブルである場合、インストラクションの特別の要求に依存して、第2のクラスのオブジェクト又は第2のクラス自身を使用することができる。

【0231】第1のプレイスを占有しているエージェントは、そのエージェントの1以上のクローンを作りだし、各々のクローンをそれぞれのプレイスに移動させることもできる。クローンは、現存のエージェントプロセスの複製の結果としてのエージェントプロセスである。エージェントは、1以上のクローンの生成を開始し制御し、さらに、オペレーション”send”の遂行によって各々のクローンのそれぞれのプレイスプロセスへの移動を開始し制御する。エージェントは生成させるべきクローンの数を指定し、オペレーション”send”のアーギュメントとして供給される1以上のチケットをつくり出すことによって、対応するトリップを定義する。各々のチケットは、エージェントの各々のクローンによってなされるトリップを定義する。エージェントによって供給されるチケットの数は、作り出されたクローンの数を規定する。

【0232】エージェントのあるクローンが第2のプレイスに移動されると、そのクローンがつくり出された時点においてのエージェントの実効状態を最初備えている。したがってそのクローンが第2のプレイスに到達した時、クローンは、オペレーション”go”について前述したように第2のプレイスへの前記エージェントへの移

動をシュミレートするように実効を継続する。

【0233】従来の技術によれば、第1のプロセスすなわちヘッド（頭）プロセスは、同一のインターフェースを持った、すなわち、ヘッドプロセスと同一のオペレーションを一般に遂行することができたリム（手足）プロセスをつくり出した。リムプロセスは、ヘッドプロセスによって指示された以外の働きはせず、単にヘッドプロセスの指示にしたがって、遠隔のコンピュータシステムに移動することができる。

【0234】しかし、本発明によれば、クローンは自立的であり、そのクローンをつくり出したエージェントによって制御されない。一例としてクローンは後述するミーティングをとりはからうためのエージェントによるすべての試みを無視するように設計することができ、それによりエージェントとクローンは相互作用することができる。エージェントは後述するように他のエージェントと相互作用することを試みるのと同じ仕方で、エージェントのクローンと相互作用するように努めなければならない。クローンは自立的であるから、第2のプレイスを占有するクローンは、エージェントによってそのように指示されることなく、またエージェントの同意さえなしに、オペレーション“go”の遂行によって第3のプレイスに移動することができる。

【0235】従来の技術によるリムプロセスは、活性でなく、ヘッドプロセスの指示によってのみ作業したので、ヘッドプロセスとの遠隔のリムプロセスとは必然的にネットワークの通信媒体を横切って相互作用していた。したがってこの従来技術によるシステムへのパラダイムは、遠隔プロシージャコーリングに一層類縁である。対照的に、本発明にしたがって形成されたエージェントのクローンは、活性であり、自立的であり、エージェントを形成するすべてのインストラクションを具現している。

【0236】したがって、ネットワーク通信媒体を横切るいかなる相互作用も必要ではない（実際にはいかなる相互作用も許容されない）。したがって本発明のパラダイムはリモートプログラミングに一層類縁である。したがって本発明は不変性において従来技術にくらべて顕著な増大を表している。

【0237】本発明の一実施例による効率の増大は、あるエージェントのクローニングをできるだけ長く遅延させることによるオペレーション“send”の遂行において実現される。一例として、第1のクローンと第2のクローンとがそれぞれ第1のプレイスと第2のプレイスとに送られるものとする。エージェントが第1のコンピュータシステムにおいて実行され、第1のプレイスが第2のコンピュータシステムにおいて実行され、第2のプレイスが第3のコンピュータシステムにおいて実効されているものと想定する。さらに、第1のプレイス及び第2のプレイスに移動する際に第1及び第2のクローンが、第

4のコンピュータシステムにおいて作動しているエンジンを通して移動しなければならないものと想定する。このような場合、単一のクローンが形成され、第4のコンピュータシステム内において実行しているエンジンに転送される。したがって元のエージェントを解釈しているエンジン内では単一のクローンがつくり出されるにすぎないので、そのエンジン内においてスペースが節減され、ただ1つのクローンがネットワーク通信媒体を横切って第4のコンピュータシステムに転送されるので、クローンを第4のコンピュータシステムに転送する時間が節減される。第4のコンピュータシステム内において実行しているエンジンは、第1及び第2のクローンを前記単一のクローンから形成し、これらの第1及び第2のクローンをそれぞれ第1及び第2のプレイスに転送する。

【0238】エンジンは、エージェントの大きさの主要部分を占めるオブジェクトたとえば、デジタル化音声のファクシミリ伝送のようなラスター化されたグラフィックイメージを所有することができるので、各々が個々のクローンによって所有されている非常に大きなオブジェクトのいくつかのコピーを単一のエンジンに送ることを避けることによって実質的な時間及び費用が節減される。

【0239】あるプレイスを占有している第1のエージェントは、その第1のエージェントとその場所を占有している第2のエージェントとの間のミーティングを開始することができる。このミーティングの間に第1のエージェントは、オブジェクトの形のデータを第2のエージェントに対して移送させたり受けたりすることができる。そして第2のエージェントプロセスは、オブジェクトの形のデータを第1のエージェントに対して移送したり受けたりすることができる。

【0240】本発明は第1のプロセスによって仮想的なブレッチンボード上においてメッセージを郵送することを教示した従来技術のシステムに対する顕著な改良を表している。メッセージは次に予定された受領プロセスによって読まれる。従来技術のシステムによれば、第1のプロセスは、仮想的なブレッチンボード上においてリファレンスを郵送することによって第1のプロセスへのリファレンスを第2のプロセスに与える。しかしリファレンスを同時に交換するメカニズムは存在しないため、第2のプロセスは、第2のプロセスへのリファレンスを第1のプロセスに与える前に第1のプロセスにアクセスする。したがって、悪意のあるプロセスがそれ自身に対するアクセスを他のプロセスに許容することなく他のプロセスへのアクセスを取得することを妨げるメカニズムは存在しない。

【0241】本発明は、第1のプロセスへのアクセスを第2のプロセスに同時に許容することなしに第1のプロセスが第2のプロセスにアクセスすることができないように2つのプロセスの間のコンタクトを確立するための

改良された方法を表している。

【0242】本発明の一実施例によれば、2つのエージェントは、両方のエージェントが同一のプレイスを占有し、そのプレイスがミーティングプレイスである場合にのみ相互作用することができる。ミーティングプレイスは、プレイスクラスのサブクラスであるミーティングプレイスクラスの1つのメンバであるプレイスである。第1のエージェントは、ミーティングプレイスがミーティングを整えるように指示するインストラクションを送出することによって、第1のエージェントと第2のエージェントとの間のミーティングを取り決めるように指示する。送出されたインストラクションはオペレーション”meet”と呼ばれ、第1のエージェントがこのインストラクションを送出することは、リクエストイングアミーティング（ミーティングの要求）と呼ばれる。

【0243】第1のエージェントは、インストラクションに対するアーギュメントとして、ミーティングを定義するペティションを供給する。ペティションは、第2のエージェントをペティションされるエージェントとして指定することによってミーティングを定義する。第2のエージェントは第2のエージェントのネーム及び／又はクラスを指定することによって指定される。ペティションはさらに、ミーティングが取り計られたり放棄されたりする時間の量を指定することによってミーティングをさらに定義する。

【0244】ミーティングを取り計らう際に、ミーティングプレイスは、第1のエージェントのネーム及びクラスを第2のエージェントに供給するとともに、第2のエージェントとそのミーティングを要求するインストラクションを第1のエージェントが送出したことを指示する。第2のエージェントは第1のエージェントのネーム及びクラスを検査し、第1のエージェントとのミーティングを受け入れるか拒絶するかをミーティングプレイスに回答する。

【0245】ミーティングが拒絶された場合、第1のエージェントは、第2のエージェントが利用可能でないことの通知を受ける。ミーティングが受け入れられない場合、ミーティングプレイスは、第1のエージェントへのリファレンスを第2のエージェントに与えるとともに、第2のエージェントへのリファレンスを第1のエージェントに与える。第1のエージェントは、第2のエージェントへのリファレンスによって（i）オペレーションを遂行するように第2のエージェントに指示し、（i i）アーギュメントとしてのオブジェクトを第2のエージェントに供給し、（i i i）第2のエージェントから結果としてのオブジェクトを受け取ることができる。第2のエージェントは第1のエージェントへのリファレンスをもつので、第2のエージェントは、第1のエージェントについて同様の能力を有する。

【0246】第1のエージェント又は第2のエージェン

トは、適切な指令を送出することによって両者の間のミーティングを終了させることができる。ミーティングプレイスは、送出された指令に応答してオペレーション”part”を実行することによって、第1のエージェントに含まれる第2のエージェントへのリファレンスをボイド（無効にする）し、さらに第2のエージェントに含まれる第1のエージェントへのリファレンスをボイドし、それにより2つのエージェントの間の相互作用を終了させる。

【0247】本発明は、第2のエージェントが同意することなしには、又は第2のエージェントに第1のエージェントへのアクセスを許容することなしには、第1のエージェントが第2のエージェントにアクセスすることができないため、従来技術にくらべて顕著な改良を表している。さらに2つのエージェントは両者が同一のミーティングにプレイスを占有しない限り相互作用できないので、エージェントに対して中間レベルのセキュリティが使用可能となる。たとえば、あるエージェントは他のエージェントによってその位置が広く知られていないミーティングプレイスを占有することによって、他のエージェントからそれ自身をプロテクトすることができる。その逆に、高度に可視的であるように設計されるエージェントは、周知のミーティングプレイスを占有することができる。このようなメカニズムは従来技術においては使用することができない。

【0248】

【実施例】以下、本発明の実施例について図面を参照しながら説明する。

【0249】本発明は、エージェントクラス及びプレイスクラスを含む複数のオブジェクト指向クラスを定義し、前記オブジェクト指向クラス、該オブジェクト指向クラスのサブクラス及び go オペレーションを含む、コンピュータプロセスのためのインストラクションを形成し、該コンピュータネットワーク中のプロセッサによって前記インストラクションをインタープリートし、前記 go オペレーションに応答して、エージェントプロセスがプレイスプロセスに転送され、前記エージェントプロセスが、前記エージェントクラスの1つのメンバであり、前記プレイスプロセスは前記プレイスクラスの1つのメンバであることを特徴とする。

【0250】本発明は、前記エージェントプロセスにおいて前記 go オペレーションを形成することを特徴とする。

【0251】本発明は、前記プレイスのサブプレイスを形成し、前記サブプレイスは前記プレイスクラスの1つのメンバであり、前記プレイスは前記サブプレイスの1つのスーパープレイスであることを特徴とする。

【0252】本発明は、前記エージェントプロセスをオブジェクトのオーナーとして指定することを特徴とする。

【0253】本発明は、前記オブジェクト中にダイジェストを形成することを特徴とする。

【0254】本発明は、前記オブジェクトの代わりに前記ブレイスプロセスにおいて第2のオブジェクトを相互変換し、前記第2のオブジェクトは、前記ダイジェストと同等の第2のダイジェストを有し、前記第1のオブジェクトは前記ブレイスプロセスに転送されないことを特徴とする。

【0255】本発明は、前記ブレイスプロセスをチケット手段によって指定することを特徴とする。

【0256】本発明は、前記チケット手段を前記エージェントプロセス内において形成することを特徴とする。

【0257】本発明は、前記チケット手段中において前記ブレイスプロセスのアドレスを形成することを特徴とする。

【0258】本発明は、前記アドレスがテレアドレスであることを特徴とする。

【0259】本発明は、前記チケット手段中において前記ブレイスプロセスのネームを形成することを特徴とする。

【0260】本発明は、前記ネームがテレネームであることを特徴とする。

【0261】本発明は、クラスオブジェクトのクラスを規定し、クラスオブジェクトを形成し、前記クラスオブジェクトは、(i) クラスオブジェクトの前記クラスの1つのメンバであり、(ii) 前記オブジェクト指向クラスの選択された1つのもののサブクラスである第1のクラスを規定し、前記第1のクラスの1つのメンバであるオブジェクトを形成し、前記エージェントプロセスは、前記オブジェクトを所有し、前記エージェントプロセスを前記ブレイスプロセスに転送することが、前記オブジェクト及び前記クラスオブジェクトを前記ブレイスプロセスに転送することを含むことを特徴とする。

【0262】本発明は、エージェントクラスとブレイスクラスとを含む複数のオブジェクト指向クラスを定義し、前記オブジェクト指向クラス、前記オブジェクト指向クラスのサブクラス及び send オペレーションを含むコンピュータプロセスのためのインストラクションを形成し、前記コンピュータのネットワーク中のプロセッサにおいて前記インストラクションをインタープリートする際に、前記 send オペレーションにตอบสนองしてエージェントプロセスの1つのクローンが1つのブレイスプロセスに転送され、前記クローン及び前記エージェントプロセスは、各々前記エージェントクラスの1つのメンバであり、前記ブレイスプロセスは前記ブレイスクラスの1つのメンバであることを特徴とする。

【0263】本発明は、前記 send オペレーションを前記エージェントプロセス中において形成することを特徴とする。

【0264】本発明は、前記ブレイスのサブブレイスを

形成し、前記サブブレイスは前記ブレイスクラスの1つのメンバであり、前記ブレイスは前記サブブレイスの1つのスーパーブレイスであることを特徴とする。

【0265】本発明は、前記エージェントプロセスをオブジェクトのオーナーとして指定することを特徴とする。

【0266】本発明は、前記クロンの転送が前記プロジェクトのコピーを前記ブレイスプロセスに転送することを特徴とする。

【0267】本発明は、前記オブジェクト中にダイジェストを形成することを特徴とする。

【0268】本発明は、前記オブジェクトと異なる第2のオブジェクトを前記ブレイスプロセスにおいて前記コピーと相互変換し、前記ブレイスプロセスにおいて前記第2のオブジェクトは、前記ダイジェストと同等の第2のダイジェストを持ち、前記コピーは前記ブレイスプロセスに転送されないことを特徴とする。

【0269】本発明は、前記ブレイスプロセスをチケット手段によって指定することを特徴とする。

【0270】本発明は、前記エージェントプロセス中において前記チケット手段を形成することを特徴とする。

【0271】本発明は、前記ブレイスプロセスのアドレスを前記チケット手段中において形成することを特徴とする。

【0272】本発明は、前記アドレスがテレアドレスであることを特徴とする。

【0273】本発明は、前記チケット手段中において前記ブレイスプロセスの1つのネームを形成することを特徴とする。

【0274】本発明は、前記ネームがテレネームであることを特徴とする。

【0275】本発明は、前記 send オペレーションにตอบสนองして前記エージェントプロセスの第2のクローンを第2のブレイスプロセスに転送し、前記第2のクローンは、前記クローンとは異なり、前記エージェントクラスの1つのメンバであり、前記第2のブレイスプロセスは前記ブレイスプロセスの1つのメンバであることを特徴とする。

【0276】本発明は、前記第1のクローンを前記第1のブレイスプロセスに転送し、前記第2のクローンを前記第2のブレイスプロセスに転送するためには、前記コンピュータネットワークの複数のコンピュータの内の単一のコンピュータに前記第1及び第2のクローンを転送することを必要であることを定め、前記第1のクローンを前記単一のコンピュータに転送し、前記単一のコンピュータにおいて前記第1のクローンから前記第2のクローンを形成することを特徴とする。

【0277】本発明は、前記単一のコンピュータから前記ブレイスプロセスに前記第2のクローンを転送し、前記単一のコンピュータから前記第2のブレイスプロセス

に前記第2のクローンを転送することを特徴とする。

【0278】本発明は、エージェントクラスを含む複数のオブジェクト指向クラスを定義し、前記オブジェクト指向クラス、前記オブジェクト指向クラスのサブクラス及びmeet オペレーションを含むコンピュータプロセスのためのインストラクションを形成し、前記コンピュータネットワーク中の1つのプロセッサにおいて前記インストラクションをインタープリートし、ミーティングブレイスプロセスが前記meet オペレーションにตอบสนองして、第2のエージェントプロセスへの第1のエージェントプロセスへのアクセスを供与し、前記第1のエージェントプロセスへの前記第2のエージェントプロセスへのアクセスも供与し、さらに、前記第1及び第2のエージェントプロセスが前記エージェントクラスのメンバであることを特徴とする。

【0279】本発明は、前記ミーティングブレイスプロセスが前記複数のオブジェクト指向クラス中のブレイスクラスの1つのメンバであり、前記第1及び第2のエージェントプロセスが前記ミーティングブレイスプロセスを占有することを特徴とする。

【0280】本発明は、第2のブレイスプロセスを形成し、第2のブレイスプロセスは前記ブレイスクラスの1つのメンバであり、前記第2のブレイスプロセスは前記ミーティングブレイスプロセスの1つのサブブレイスであり、前記ミーティングブレイスは前記第2のブレイスのスーパーブレイスであることを特徴とする。

【0281】本発明は、エージェントクラス、ブレイスクラス及びチケットクラスを含む複数のオブジェクト指向クラスを定義し、各々が前記ブレイスクラスの1つのメンバである、前記コンピュータネットワーク中の複数のブレイスプロセスを形成し、前記エージェントクラスの1つのメンバであって前記複数のブレイスプロセス中の第1のブレイスプロセスを占有する、エージェントプロセスを形成し、前記チケットクラスの1つのメンバであり、前記複数のブレイスプロセス中の前記第1のブレイスプロセスから第2のブレイスプロセスへの前記エージェントプロセスの移動を含むトリップを定義するチケットを形成することを特徴とする。

【0282】本発明は、ブレイスプロセスとパーミットクラスとを含む複数のオブジェクト指向クラスを定義し、前記ブレイスクラスの1つのメンバであるプロセスを形成し、前記パーミットクラスの1つのメンバであって前記プロセスの1以上の能力を指定するパーミットを形成することを特徴とする。

【0283】本発明は、前記プロセスクラス中に go オペレーションを定義し、前記プロセスが前記 go オペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする。

【0284】本発明は、前記プロセスクラス中に send オペレーションを定義し、前記プロセスが前記 send オ

ペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする。

【0285】本発明は、前記プロセスクラス中において charge オペレーションを定義し、前記プロセスが前記 charge オペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする。

【0286】本発明は、前記プロセスクラス中に terminate オペレーションを定義し、前記プロセスが前記 terminate オペレーションを遂行しうるか否かを前記パーミット中において指定することを特徴とする。

【0287】本発明は、前記プロセスが前記プロセスと異なった第2のプロセスを作り出すことができるか否かを前記パーミット中において指定し、該第2のプロセスは前記プロセスクラスの1つのメンバであることを特徴とする。

【0288】本発明は、オブジェクト指向ブレイスクラスを定義し、前記パーミット中に前記プロセスが前記ブレイスプロセスの複数のメンバを作り出すことができるか否かを前記パーミット中において指定することを特徴とする。

【0289】本発明は、前記プロセスが失敗した時に前記プロセスが再スタートされるか否かを前記パーミット中において指定することを特徴とする。

【0290】本発明は、前記プロセスに割り当てられた処理量を前記パーミット中において指定することを特徴とする。

【0291】本発明は、前記プロセスが終了する時間を前記パーミット中において指定することを特徴とする。

【0292】本発明は、前記プロセスクラス中において制限オペレーションを定義し、前記クラス、前記クラスのサブクラス及び前記制限オペレーションを含むコンピュータプロセスのためのインストラクションを形成し、前記コンピュータネットワーク中のプロセッサにおいて前記インストラクションをインタープリートし、前記パーミットと異なる第2のパーミットが前記制限オペレーションにตอบสนองして形成され、前記第2のパーミットは前記パーミットクラスの1つのメンバであり、前記プロセスの前記1以上の能力の1つのグループを指定し、前記第2のパーミットによって指定された1以上の能力の前記グループに前記プロセスを制限することを特徴とする。

【0293】本発明は、コンピュータにおいて、クラス（複数）の1つのクラスとサイテーション（複数）の1つのクラスを含む、複数のオブジェクト指向クラスを定義し、前記複数のクラスの前記クラスのメンバである1以上のクラスオブジェクトを形成し、前記第1のクラスオブジェクトを指定するとともに、前記クラスオブジェクトの内のどれが前記第1のオブジェクトに対して後方にコンパティブルであるかを指定するサイテーションを前記クラスオブジェクト内の第1のものの中に形成する



ことを含む、インストラクションセットの種々のバージョンのプロセス（複数）をインタープリットする。

【0294】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、各々がゼロ又はそれ以上のエージェントプロセスのための前記コンピュータ内の1つのもののある場所である複数のプレイスプロセスを前記コンピュータネットワーク中に用意し、前記複数のプレイスプロセス中の目的点プレイスプロセスへのエージェントプロセスのトリップをチケット手段によって指定し、前記エージェントプロセス中の send オペレーションに回答して前記エージェントプロセスの1つのクローンを前記目的点プレイスプロセスに転送することを特徴とする。

【0295】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記複数のプレイスプロセス中の第2の目的点プレイスプロセスへの前記第2のエージェントプロセスの前記トリップと異なる第2のトリップを第2のチケット手段によって指定することを特徴とする。

【0296】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記第1のトリップを行なう前記クローンと前記第2のトリップを行なう第2のクローンとがどちらも前記コンピュータの内の単一のコンピュータに転送されるべきことを定め、前記第1のクローンを前記単一のコンピュータに移動させ、前記単一のコンピュータ中に、前記第1のクローンと異なる前記第2のクローンを形成することを特徴とする。

【0297】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記第1のクローンを前記目的点プレイスプロセスに転送することを特徴とする。

【0298】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記第2のクローンを前記第2の目的点プレイスプロセスに転送することを特徴とする。

【0299】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェントプロセス中の go オペレーションに回答して、前記チケット手段によって指定される前記目的点プレイスプロセスに前記エージェントプロセスを転送することを特徴とする。

【0300】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段中のネームによって前記目的点プレイスプロセスを指定することを特徴とする。

【0301】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記ネームがテレネームであることを特徴とする。

【0302】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段中

のアドレスによって前記目的点プレイスプロセスを指定することを特徴とする。

【0303】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記アドレスがテレアドレスであることを特徴とする。

【0304】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点プレイスプロセスがその1つのメンバであるクラスのサイテーションによって前記目的点プレイスプロセスを前記チケット手段中において指定することを特徴とする。

【0305】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点プレイスプロセスのアドレス、前記目的点プレイスプロセスのネーム及び前記目的点プレイスプロセスがその1つのメンバであるクラスのサイテーションの任意の組み合わせによって前記目的点プレイスプロセスを前記チケット手段中において指定することを特徴とする。

【0306】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記トリップの最大の期間を前記チケット手段中に指定することを特徴とする。

【0307】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記トリップのための望ましい期間を前記チケット手段中に指定することを特徴とする。

【0308】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、転送手段を前記チケット手段中に指定し、該転送手段は前記クローンを転送するためのコンピュータ間通信手段の形式を指定することを特徴とする。

【0309】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点プレイスプロセスにおいて前記エージェントプロセスのデッドラインを、前記チケット手段中に含まれる目的点パーミットによって制御することを特徴とする。

【0310】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点プレイスプロセスにおいて前記エージェントプロセスが遂行することが許可されたオペレーションを、前記チケット手段に含まれる目的点パーミットによって制御することを特徴とする。

【0311】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点プレイスプロセスにおいて前記エージェントプロセスが消費することが許可されたリソースを、前記チケット手段に含まれる目的点パーミットによって制御することを特徴とする。

【0312】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点プレイスプロセスにおいての他のプロセスに対する前記目的点プ



レイスプロセスにおいての前記エージェントプロセスの相対的なプライオリティを、前記チケット手段に含まれる目的点パーミットによって指定することを特徴とする。

【0313】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、あるプロセスが持つことを許可される能力をパーミット手段を介して制御することを特徴とする。

【0314】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段が固有のパーミットであることを特徴とする。

【0315】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段がローカルパーミットであることを特徴とする。

【0316】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段が一時的なパーミットであることを特徴とする。

【0317】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が前記プロセスの消費しうるリソースからなることを特徴とする。

【0318】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、その後ではプロセスが進行しえないデッドラインを含むことを特徴とする。

【0319】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が他のプロセスに対する前記プロセスの相対的なプライオリティを含むことを特徴とする。

【0320】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、前記プロセスが選択されたオペレーションを遂行する許可を含むことを特徴とする。

【0321】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェントプロセスがオブジェクトを所有することを特徴とする。

【0322】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記オブジェクトがダイジェストを有することを特徴とする。

【0323】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェントプロセスのクローンを転送する前記工程が前記オブジェクトのコピーを転送することを含むことを特徴とする。

【0324】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記コピーを第2のオブジェクトに相互変換し、該第2のオブジェクトは、最初に述べた前記ダイジェストと同等のダイジェストを有し、前記コピーは前記目的点ブレイスプロセスには転送されないことを特徴とする。

【0325】本発明は、複数のコンピュータを有するコ

ンピュータネットワークにおいて、前記目的点ブレイスプロセスにおいて前記オブジェクトの代わりに、相互変換されたオブジェクトを使用して、前記オブジェクトを前記目的点ブレイスプロセスに転送する必要を除くことを特徴とする。

【0326】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記転送工程が前記目的点ブレイスプロセスにエンタする工程を含むことを特徴とする。

【0327】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記転送ステップが前記ソースブレイスプロセスをエクシットする工程を含むことを特徴とする。

【0328】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイスプロセスがミーティングブレイスプロセスであり、前記ミーティングブレイスはリクエストエージェントプロセスとペティションドエージェントプロセスとの間のミーティングを取り計らうことを特徴とする。

【0329】本発明は、コンピュータにおいて、第1のエージェントプロセスと第2のエージェントプロセスとを用意し、ペティション手段によって、前記第1のエージェントプロセスと第2のエージェントプロセスとの間のミーティングを指定し、前記ペティション手段によって定義された前記第1のエージェントプロセスと第2のエージェントプロセスとの間の前記ミーティングを取り計らうことを特徴とする。

【0330】本発明は、コンピュータにおいて、前記第1のエージェントプロセス中に前記ペティション手段を形成することを特徴とする。

【0331】本発明は、コンピュータにおいて、ミーティングを特定する前記工程は、前記ペティション手段中において前記第2のエージェントプロセスを指定することを特徴とする。

【0332】本発明は、コンピュータにおいて、前記第2のエージェントプロセスを指定する前記工程が、前記第2のエージェントをネームによって前記ペティション手段中において指定することを特徴とする。

【0333】本発明は、コンピュータにおいて、前記ネームがテレネームであることを特徴とする。

【0334】本発明は、コンピュータにおいて、前記第2のエージェントプロセスを指定する前記工程が、前記第2のエージェントプロセスがその1つのメンバであるクラスを前記ペティション手段中において指定することを特徴とする。

【0335】本発明は、コンピュータにおいて、前記第2のエージェントプロセスを指定する前記工程は、前記第2のエージェントプロセスがその1つのメンバであるクラスのサイテーションを前記ペティション手段中において指定することを特徴とする。

【0336】本発明は、コンピュータにおいて、ミーティングを指定する前記工程が、前記ミーティングを取り計らうための最大の期間を前記ベティション手段中において指定することを特徴とする。

【0337】本発明は、コンピュータにおいて、前記ミーティングを取り計らう前記工程は、前記第2のエージェントプロセスへのリファレンスを前記第1のエージェントプロセスに供与することを特徴とする。

【0338】本発明は、コンピュータにおいて、前記ミーティングを取り計らう前記工程が、前記第1のエージェントプロセスへのリファレンスを前記第2のエージェントプロセスに対して供与することを特徴とする。

【0339】本発明は、コンピュータにおいて、前記第2のエージェントプロセス中のオペレーションの遂行を前記第1のエージェントプロセス中においてインスタクションに应答して生じさせることを特徴とする。

【0340】本発明は、コンピュータにおいて、前記第1のエージェントプロセスがオブジェクトを所有することを特徴とする。

【0341】本発明は、コンピュータにおいて、前記オブジェクトへのリファレンスを前記第2のエージェントプロセスに対して供与することを特徴とする。

【0342】本発明は、コンピュータにおいて、前記リファレンスが保護されるリファレンスであることを特徴とする。

【0343】本発明は、コンピュータにおいて、前記オブジェクトのコピーへのリファレンスを前記第2のエージェントプロセスに供与することを特徴とする。

【0344】本発明は、コンピュータにおいて、前記第2のエージェントプロセスがオブジェクトを所有することを特徴とする。

【0345】本発明は、コンピュータにおいて、前記オブジェクトへのリファレンスを前記第1のエージェントプロセスに対して供与することを特徴とする。

【0346】本発明は、コンピュータにおいて、前記リファレンスがプロテクトされたリファレンスであることを特徴とする。

【0347】本発明は、コンピュータにおいて、前記オブジェクトのコピーへのリファレンスを前記第1のエージェントプロセスに対して供与することを特徴とする。

【0348】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、チケット手段及び send オペレーションを有するエージェント手段と、前記複数のコンピュータの1つにおいてその各々が動作する複数のブレイス手段とを有し、前記エージェント手段は前記複数のブレイス手段中の第1のブレイス手段であり、前記チケット手段は、前記複数のブレイス手段中の目的点ブレイス手段への前記エージェント手段のトリップを指定し、前記 send オペレーションは、前記エージェント手段の1つのクローンを前記目的点手段に転送す

ることを特徴とする。

【0349】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段が、前記エージェント手段のための第2の目的点ブレイス手段への第2のトリップを指定する、前記チケット手段と異なる第2のチケット手段を、含むことを特徴とする。

【0350】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記 send オペレーションの遂行によって前記エージェント手段の第2のクローンが前記第2の目的点ブレイス手段に転送されることを特徴とする。

【0351】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記トリップ及び前記第2のトリップが、前記複数のコンピュータ中のある単一のコンピュータへの転送を含み、前記クローンを前記目的点ブレイス手段に転送し、前記第2のクローンを前記第2の目的点手段に転送する際に、前記 send オペレーションの遂行によって、(a) 前記第1のクローンが前記単一のコンピュータに転送され、(b) 前記単一のコンピュータ中において前記第1のクローンから前記第2のクローンが形成されることを特徴とする。

【0352】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段が go オペレーションを含み、前記 go オペレーションは、第3のチケット手段によって指定された第3の目的点ブレイス手段に前記エージェント手段を転送することを特徴とする。

【0353】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記目的点ブレイス手段を特定するネームを含むことを特徴とする。

【0354】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記ネームがテレネームであることを特徴とする。

【0355】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記目的点ブレイス手段のためのアドレスを含むことを特徴とする。

【0356】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記アドレスがテレアドレスであることを特徴とする。

【0357】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記目的点ブレイス手段をその1つのメンバとするクラスを指定するサイテーション手段を含むことを特徴とする。

【0358】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が、前記目的点ブレイス手段のアドレスと、前記目的点

ブレイス手段のネームと、前記目的点ブレイス手段をその1つのメンバとするクラスを指定するサイテーション手段とからなる群中より選ばれた任意の組合せを含むことを特徴とする。

【0359】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記トリップのための最大の期間を指定する手段を含むことを特徴とする。

【0360】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が前記トリップのための望ましい期間を指定する手段を含むことを特徴とする。

【0361】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が、前記トリップ手段を完成させるコンピュータ間通信手段の形式を指定するウェイ手段を含むことを特徴とする。

【0362】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記チケット手段が、前記目的点ブレイス手段においての前記エージェント手段のためのデッドラインを制御するパーミット手段を含むことを特徴とする。

【0363】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段が前記目的点ブレイス手段において遂行することが許可されているオペレーションを制御するためのパーミット手段を前記チケット手段を有することを特徴とする。

【0364】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイス手段において前記エージェント手段が消費することが許可されているリソースを制御するためのパーミット手段を前記チケット手段が有することを特徴とする。

【0365】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイス手段においての他のエージェント手段に対する前記目的点ブレイス手段においての前記エージェント手段の相対的なプライオリティを指定するためのパーミット手段を前記チケット手段が有することを特徴とする。

【0366】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、エージェント手段が持つことが許可される能力を制御するためのパーミット手段を有することを特徴とする。

【0367】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段が固有のパーミットであることを特徴とする。

【0368】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段がローカルパーミットであることを特徴とする。

【0369】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記パーミット手段

が一時的なパーミットであることを特徴とする。

【0370】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が前記エージェント手段の消費するリソースを含むことを特徴とする。

【0371】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、前記エージェント手段がそれから後進行することのできないデッドラインを含むことを特徴とする。

【0372】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、他のエージェント手段を含み、前記能力は、前記他のエージェント手段に対する前記エージェント手段の相対的なプライオリティを含むことを特徴とする。

【0373】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記能力が、前記エージェント手段が選択されたオペレーションを遂行する許可を含むことを特徴とする。

【0374】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記エージェント手段がオブジェクトを所有することを特徴とする。

【0375】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記オブジェクトがダイジェストを有することを特徴とする。

【0376】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記オブジェクトのコピーが前記クローンとともに、前記 send オペレーションの遂行によって転送されることを特徴とする。

【0377】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記目的点ブレイス手段において第2のオブジェクトは前記コピーに相互変換され、前記第2のオブジェクトは、前記ダイジェストと同等のダイジェストを持つことによって前記コピーを前記目的点ブレイス手段に転送することを不要とすることを特徴とする。

【0378】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記 send オペレーションが前記目的点ブレイス手段においての前記コピーの代わりに相互変換されたオブジェクトを使用することによって、前記オブジェクトの前記コピーを前記目的点ブレイス手段に転送する必要を除くことを特徴とする。

【0379】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記ブレイス手段は前記ブレイス手段にエンタするための手段を有することを特徴とする。

【0380】本発明は、複数のコンピュータを有するコンピュータネットワークにおいて、前記ブレイス手段が前記ブレイス手段をエグジットする手段を有することを特徴とする。

【0381】本発明は、複数のコンピュータを有するコ

ンピュータネットワークにおいて、前記ブレイス手段がミーティングブレイス手段を含み、前記ミーティングブレイス手段はリクエストエージェント手段とレスポングエージェント手段との間のミーティングを取り計らうことを特徴とする。

【0382】本発明は、コンピュータにおいて、meetオペレーションを有するミーティングブレイス手段を有し、該ミーティングブレイスは、複数のエージェント手段のための場所であり、更に、ペティッション手段と、前記複数のエージェント手段の内の第1のエージェント手段と、前記複数のエージェント手段の中の第2のエージェント手段とを有し、前記第1のエージェント手段及び第2のエージェント手段は前記複数のエージェント手段の内のどれともミートでき、前記ペティッション手段は前記第1のエージェント手段と第2のエージェント手段との間のミーティングを指定し、前記ミーティング手段は前記meetオペレーションに応答して前記ミーティングを取り計らうことを特徴とする。

【0383】本発明は、コンピュータにおいて、前記第1のエージェント手段が前記ペティッション手段を有することを特徴とする。

【0384】本発明は、コンピュータにおいて、前記ペティッション手段が前記第2のエージェントプロセスを規定することによって前記ミーティングを指定することを特徴とする。

【0385】本発明は、コンピュータにおいて、前記ペティッション手段が前記第2のエージェント手段を指定するネームを含むことを特徴とする。

【0386】本発明は、コンピュータにおいて、前記ネームがテレネームであることを特徴とする。

【0387】本発明は、コンピュータにおいて、前記第2のエージェントプロセスがその1つのメンバであるクラスを指定することによって前記ペティッション手段が前記第2のエージェントを指定することを特徴とする。

【0388】本発明は、コンピュータにおいて、前記ペティッション手段がサイテーションを含み、該サイテーションは、前記第2のエージェントプロセスがその1つのメンバであるクラスを指定することを特徴とする。

【0389】本発明は、コンピュータにおいて、前記ペティッション手段が前記ミーティングを取り計らうための最大の期間を規定することを特徴とする。

【0390】本発明は、コンピュータにおいて、前記meetオペレーションが前記第2のエージェント手段へのリファレンスを前記第1のエージェント手段に供与することを特徴とする。

【0391】本発明は、コンピュータにおいて、前記meetオペレーションが前記第1のエージェント手段へのリファレンスを前記第2のエージェント手段に供与することを特徴とする。

【0392】本発明は、コンピュータにおいて、前記第

1のエージェント手段が前記第2のエージェント手段中のオペレーションの遂行を生じさせることを特徴とする。

【0393】本発明は、コンピュータにおいて、前記第1のエージェント手段がオブジェクトを所有することを特徴とする。

【0394】本発明は、コンピュータにおいて、前記第1のエージェント手段が前記オブジェクトへのリファレンスを前記第2のエージェント手段に供与することを特徴とする。

【0395】本発明は、コンピュータにおいて、前記リファレンスがプロテクトされたリファレンスであることを特徴とする。

【0396】本発明は、コンピュータにおいて、前記第1のエージェント手段が前記オブジェクトのコピーへのリファレンスを前記第2のエージェント手段に供与することを特徴とする。

【0397】本発明は、コンピュータにおいて、前記第2のエージェント手段がオブジェクトを所有することを特徴とする。

【0398】本発明は、コンピュータにおいて、前記第2のエージェント手段が前記オブジェクトへのリファレンスを前記第1のエージェント手段に供与することを特徴とする。

【0399】本発明は、コンピュータにおいて、前記リファレンスがプロテクトされたリファレンスであることを特徴とする。

【0400】本発明は、コンピュータにおいて、前記第2のエージェント手段が前記オブジェクトのコピーへのリファレンスを前記第1のエージェント手段に供与することを特徴とする。

【0401】本発明は、1以上のコンピュータを有するコンピュータネットワークにおいて、第1のエンジンプロセスから第2のエンジンプロセスにデータを移送するデータ移送方法において、(a)前記第1のエンジンプロセス及び第2のエンジンプロセスを含む1以上のエージェントプロセスを実行するための実行手段を用意し、各々の前記エージェントプロセスは、コンピュータインストラクションセットからの複数のインストラクションを含み、実行状態を持つものとし、(b)前記コンピュータインストラクションセット中にgoインストラクションを用意し、該goインストラクションは、前記第1のエンジンプロセス中において実行される第1のエージェントプロセスに含まれるものとし、前記goインストラクションの遂行によって、(i)前記第1のエンジンプロセスによる前記第1のエージェントプロセスの実行の中断と、(ii)前記第1のエージェントプロセスの前記実行状態が保存されるような前記第1のエージェントプロセスの表現と、(iii)前記第1のエンジンプロセスから第2のエンジンプロセスへの前記第1のエ

ジェントプロセスの前記表現の移送と、(i v) 前記第2のエンジンプロセスによる前記第1のエージェントプロセスの実行の再開とを生じさせ、(c) 前記第1のエージェントプロセスによる前記エージェントプロセスの実行を生じさせることによって前記 go インストラクションの遂行を生じさせることを特徴とする。

【0402】本発明は、前記コンピュータインストラクションセットがオブジェクト指向であることを特徴とする。

【0403】本発明は、前記複数のエージェントプロセスが前記オブジェクト指向コンピュータインストラクションセットのオブジェクトであることを特徴とする。

【0404】本発明は、前記 go インストラクションの遂行を生じさせる前に前記第1のエージェントプロセスにデータを付加し、前記第1のエージェントプロセスの前記表現が前記データを含み、前記第2のエンジンプロセスへの前記第1のエージェントプロセスの前記表現の移送が前記データを移送を含むことを特徴とする。

【0405】本発明は、前記第1のエージェントプロセスが、前記第1のエンジンプロセスから前記第2のエンジンプロセスに転送されるべきメッセージを表すデータを含み、前記 go インストラクションの実行によって生ずる前記第1のエージェントプロセスの移送が前記第1のエンジンプロセスから前記第2のエンジンプロセスへの前記データの移送を生じさせることを特徴とする。

【0406】本発明は、前記第1のエンジンプロセスが、第1のコンピュータにおいて実行され、前記第2のエンジンプロセスが第2のコンピュータによって実行され、前記第1及び第2のコンピュータが前記コンピュータネットワークの一部分であることを特徴とする。

【0407】本発明は、第1のエンジンプロセスから1以上のエンジンプロセスにデータを移送するデータ移送方法において、(a) コンピュータインストラクションセットからのインストラクションを含む複数のエージェントプロセスを実行するための手段を用意し、該エージェントプロセスを実行する手段が、前記第1のエンジンプロセス及び1以上のエンジンプロセスを含み、各々の該エージェントプロセスは実行状態を含み、(b) 前記コンピュータインストラクションセット中に send インストラクションを用意し、該 send インストラクションは、前記複数のエージェントプロセス中の第1のエージェントプロセス中に含まれるようにし、前記 send インストラクションの実行は、(i) 前記第1のエージェントプロセスの1以上のコピーを形成し、これらのコピーが、前記第1のエージェントプロセスの前記実行状態を保存するとともにそれを含み、(i i) 前記第1のエージェントプロセスのコピーの各々を前記第1のエンジンプロセスから前記1以上のエンジンプロセスのそれぞれに移送し、(i i i) 前記第1のエージェントプロセスの前記コピーの各々を前記1以上のエンジンプロセスのそれぞ

れの1つによって実行して前記第1のエージェントプロセスの再開された実行をシュミレートすることを特徴とする。

【0408】本発明は、前記第1のエージェントプロセスの前記コピーの2以上が前記第1のエンジンプロセスから単一の第2のエンジンプロセスに移送されるべき条件のもとにおいて、前記第1のエージェントプロセスの前記2以上のコピーの移送が、前記第1のエージェントプロセスの単一のコピーを前記第2のエンジンプロセスに移送する工程と、前記第2のエンジンプロセス中において前記単一のコピーから、前記第1のエージェントプロセスの前記2以上のコピーを形成する工程とからなることを特徴とする。

【0409】本発明は、あるコンピュータシステムにおいて実行されている第1のエージェントプロセスから、該コンピュータシステムによって実行されている第2のエージェントプロセスにデータを転送する際に、前記第1のエージェントプロセス及び第2のエージェントプロセスがあるブレイスプロセスの占有者であるようにしてデータを転送するデータ転送方法において、前記第1のエージェントプロセスが送出するミートインストラクションに回答して、前記第2のエージェントプロセスによるプロシージャの実行を生じさせる工程を含み、前記プロシージャは、前記第2のエージェントプロセスの一部であり、前記第2のエージェントコンピュータプロセスに含まれるコンピュータインストラクションのコレクションを含み、前記第2のエージェントプロセスによって送出される、前記プロシージャの一部である第2のインストラクションに回答して、前記第2のエージェントプロセスへのアクセス手段を前記第1のエージェントプロセスに供与する工程を含むことを特徴とする。

【0410】本発明は、前記第2のエージェントプロセスによるプロシージャの実行を生じさせる前記工程は、前記第1のエージェントプロセスによって送出される前記ミートインストラクションに回答して前記ブレイスプロセスによる第2のプロシージャの実行を生じさせることを含み、前記ブレイスプロセスによる前記第2のプロシージャの実行が、前記プロシージャを前記第2のエージェントプロセスによって実行させるインストラクションを送出することを特徴とする。

【0411】本発明は、前記第2のエージェントプロセスへのアクセス手段が前記ブレイスプロセスによって前記第1のエージェントプロセスに供与されることを特徴とする。

【0412】本発明は、近似的に、前記ブレイスプロセスが前記第2のエージェントプロセスへの前記アクセス手段を前記第1のエージェントプロセスに供与する時点において、前記ブレイスプロセスが前記第1のエージェントプロセスへのアクセス手段を前記第2のエージェントプロセスに供与することを特徴とする。

【0413】本発明は、第1のCPU及び第1のメモリを含む第1のコンピュータシステムから、第2のCPU及び第2のメモリを含む第2のコンピュータシステムへ第1のコンピュータプロセスを移送するプロセスの移送方法において、ある実行状態を持つ前記第1のコンピュータプロセスの実行を前記第1のCPUにおいて開始し、前記第1のCPU内において前記第1のコンピュータプロセスの実行を中断し、前記第1のコンピュータプロセスをデータとして前記第1のメモリ中に表現し、前記データは前記第1のコンピュータプロセスの実行が中断された時点においての前記第1のコンピュータプロセスの前記実行状態を含むものとし、前記データを前記第1のメモリから前記第2のメモリに移送し、前記データ中に表現された前記実行状態を有する第2のコンピュータプロセスを前記データから前記第2のコンピュータシステム上に形成し、前記第2のコンピュータプロセスを実行することによって前記第2のCPU内において前記第1のコンピュータシステムの実行の再開を実効的にシミュレートすることの特徴とする。

【0414】ここで、この実施例において使用する用語について説明する。

【0415】アブストラクトクラス (abstract class) : 抽象的で例 (インスタンス) を持たないクラス。アブストラクトクラスは、サブクラスを持つことができ、アブストラクトクラスは、アブストラクトクラスのサブクラスによって引き継がれたフィーチャ (feature)、方法及びプロパティ (property) を定義することができる。

【0416】エージェント (agent) : エージェントは、あるプレイスをしめ、可動である、すなわち第1のプレイスから第2のプレイスに移動することのできるプロセスである。

【0417】アーギュメント (arguments) : アーギュメントは、入力データとしてのあるオペレーションの遂行によって”消費”されるオブジェクトである。換言すれば、アーギュメントは、リクエストの要求によりレスポンドにおいてオペレーションが遂行される直前においてリクエストからレスポンドに転送されるオブジェクトである。

【0418】属性 (attribute) : 属性は、あるオブジェクトの内部状態に関する情報を検索し又はセットするフィーチャである。通常情報はオブジェクト自身に所属するが、時には、情報は、属性を呼び出す際にオブジェクトを特定化するためのリファレンス (参照) に関する。属性は一对のオペレーションであり、一方のオペレーションはレスポンドの内部状態に関する情報をセットし他のオペレーションは、レスポンドの内部状態に関する情報を検索する。

【0419】オーソリティ (authority) : オーソリティは、ネットワークの種々のリソースを所有し管理する

エンティティである。オーソリティの一例はネットワークのユーザーである。オーソリティはアドミニストラティブに作りだされ、プログラムによっては作り出されない (すなわちプロセスの要求によっては作り出されない)。

【0420】クラス (class) : クラスは (i) クラスのメンバの内部状態を規定するゼロ又はそれ以上のプロパティ、(ii) クラスのメンバの内部挙動を規定するゼロ又はそれ以上の方法、及び (iii) クラスのメンバの外部的な挙動を規定するゼロ又はそれ以上のフィーチャ、を定義する。

【0421】コンクリートクラス (concrete class) : コンクリート (具体的) であり例を有するクラス。

【0422】エンジンプレイス (engine place) : 各々のエンジンは、エンジン自身を表わす正確に1つのエンジンプレイスを収容する。

【0423】エンジン (engine) : エンジンは、オブジェクト、第一にはプロセス、を管理しインストラクション (命令) を実行するコンピュータシステム内のマシンである。エンジンは、典型的には、オペレーティングシステム及び種々のユーザーアプリケーションの他に、コンピュータシステム内において実行されるコンピュータプロセスである。1以上のエンジンは、ネットワークの各々のコンピュータシステム内において実行することができる。各々のエンジンは少なくとも1のプレイスを処理する。

【0424】エクセプション (exception) : エクセプション (例外) は、あるフィーチャが完全に、又は成功のうちに実行されることができなかった場合にそのフィーチャの遂行によって”投出 (スロー)”されるオブジェクトである。ここに使用される用語としてのエクセプションは、別途には、このようなオブジェクトを放出させる状態である。エクセプションはあるオペレーションの失敗によって生じ従って成功したオペレーションによって生じた結果とは異なっているため、レスポンドは、エクセプションを”生産”するのではなくエクセプションを”投出”すると言われる。ある結果の生産と例外の放出との区別は、アペンディックスAにおいて一層詳細に説明される。このアペンディックスAは参照によって全体としてこの明細書の一部とされる。

【0425】フィーチャ (feature) : フィーチャはあるオブジェクトが遂行するように指示されることのできるタスクである。タスクは、一組のコンピュータインストラクションを含む方法によって遂行される。あるフィーチャの遂行は、そのフィーチャの方法のコンピュータ指令の実行によって達成される。あるフィーチャは、ある特定のオブジェクトのクラスと関係があり、あるフィーチャの遂行は、そのフィーチャを遂行するオブジェクトの特定の内部状態と共に変化しうる。フィーチャは、概念的には、次の2つのカテゴリすなわち (i) 属

性及び (i i) オペレーションに分類される。

【0426】フレーム (frame) : フレームは、フィーチャの遂行の間そのフィーチャを具体化 (インプリメント) する方法の動的状態を記録するオブジェクトである。フレームは、エンジンが実行している方法に関する情報 (現に実行されている特別のインストラクション及び前記方法によって具体化されるフィーチャを遂行するオブジェクトを特定化する情報を含む) を保持するためにエンジンによって使用される。

【0427】識別子 (identifier) : 識別子 (アイデンティファイ) は、第2のオブジェクトを参照しうるオブジェクトである。識別子の "テキスト" は、ある特別の範囲内において他の識別子から識別子を区別するストリング (文字列) である。識別子のいろいろの範囲はアペンディックスAに詳細に説明されている。

【0428】インプリメンテーション (implementation) : インプリメンテーションは、ある特別のフィーチャの遂行において実行されるコンピュータステップのセットである。インプリメンテーションオブジェクト (implementation object) は、あるクラスのいろいろのフィーチャのいろいろのインプリメンテーションを定義するオブジェクトである。

【0429】インスタンス (instance) : オブジェクトは、オブジェクトがあるクラスのメンバであり、そのクラスのいかなるサブクラスのメンバでもない場合にそのクラスのインスタンス (例) である。

【0430】インターフェイス (interface) : インターフェイスは、特別の属性又は、消費された特別のアーギュメントと、ある特別のオペレーションの遂行によって生じた結果とを定義する。インターフェイスオブジェクト (interface object) は、あるクラスの種々のフィーチャの種々のインターフェイスを定義するオブジェクトである。

【0431】メンバ (member) : あるオブジェクトは、それが1つのインスタンスであるクラスのメンバであり、又そのクラスのいかなるスーパークラスのメンバでもある。

【0432】方法 (method) : 方法は、ある特別のフィーチャの遂行をその実行が構成するところのコンピュータインストラクションの1セットである。"方法オブジェクト" は、1つの方法を規定するオブジェクトである。ある方法は、その方法の実行の間すなわちその方法のインストラクションの実行の間、ある動的な状態をもつ。ある方法の動的な状態はフレームによって表わされる。

【0433】ネットワーク (network) : 全てのエンジンブレイス、それらのエンジンブレイスが実行されるコンピュータシステム、及びこれらのコンピュータシステムを接続する通信装置は、全体として集合的に1つのネットワークを形成する。

【0434】オブジェクト (object) : オブジェクトは、コンピュータシステム内のコンピュータ環境内の1つのエレメントである。オブジェクトは、ゼロ又は1以上のセイジョウによって規定される内部状態、ゼロ又は1以上の方法によって規定される内部挙動及びゼロ又はそれ以上のフィーチャによって規定される外部挙動を有する。

【0435】オペレーション (operation) : オペレーションは、インターフェイス及びインターインプリメンテーションが規定された1つのフィーチャである。

【0436】プレイス (place) : プレイスは、ゼロ又はそれ以上の方法の場所 (locale) であるプロセスである。あるプレイスは第2のプレイスを占有しうる。初めに述べたプレイスは、第2のプレイスのサブプレイスであり、第2のプレイスは第1のプレイスのスーパープレイスである。

【0437】プリミティブ (primitive) : プリミティブは、プロシージャ又は方法の形成において使用しうるオブジェクトすなわちインストラクションとして用いられるオブジェクトである。

【0438】プロセス (process) : プロセスは、自律的な計算を構成するオブジェクトである。プロセスは別のオブジェクトによってそのように要求されえることなしに方法を遂行するので、自律的である。あるプロセスは、そのプロセスの創設と共に中央方法の遂行を開始し、中央方法の完成と共にそのプロセスは破壊される。各々のオブジェクトは正確に1つの方法によって所有される。各々のプロセスはそれ自身によって所有される。この明細書で使用する "コンピュータプロセス" と言う用語は、より一般的な周知の定義である、コンピュータシステムによって遂行される一連のインストラクションを意味する。

【0439】プロパティ (property) : プロパティは、第2のオブジェクトの内部状態の一部を表わすオブジェクトである。

【0440】リファレンス (reference) : リファレンス (参照) は、特別のオブジェクトを特定化するデータ構造である。あるフィーチャを遂行するように指示されたオブジェクトは、フィーチャのリクエストによって供給されたオブジェクトへの参照によって特定化される。参照はプロテクトされていることもありプロテクトされていないこともある。オブジェクトは、それを特定化するために、プロテクトされた参照を用いて、変更され、又は改変されることができる。

【0441】リージョン (region) : リージョン (領域) は、単一のオーソリティによって管理されるネットワーク内の1以上のエンジンブレイスである。領域は、その領域のエンジンブレイス (複数) を支持するコンピュータシステムの厳密な連係化及び管理によって一般に識別される。従ってある領域内のエージェントの移送



は、複数の領域の間のエージェントの移送に比べて一般に速やかであり又コストも低廉になる。ある領域は、一例として、広い領域のネットワークに接続された局所的な領域のネットワークでありうる。

【0442】リクエスタ (requester) : リクエスタ (要求者) は、別のオブジェクトすなわちレスポнда (回答者) にあるフィーチャを遂行するように指示するオブジェクトである。リクエスタは要求されたフィーチャのレスポндаに入力データとしてゼロ又はそれ以上のオブジェクトを供給し、そのフィーチャのレスポндаから出力データとしてゼロ又は1つのオブジェクトを受け取る。あるフィーチャを遂行するようにあるオブジェクトに指示することは別途には、そのフィーチャを遂行するようにオブジェクトに”リクエスト” (要求) するととも呼ばれる。

【0443】レスポнда (responder) : レスポнда (回答者) は、別のオブジェクトすなわちリクエスタの指示に従ってあるフィーチャを実行するオブジェクトである。レスポндаは入力データとしてゼロ又は1つのオブジェクトを受け、そのフィーチャの遂行にあたってゼロ又は1つのオブジェクトを出力データとして供給する。レスポндаは別途には、”レスポンドする” オブジェクトとも呼ばれる。

【0444】結果 (result) : 結果は、出力データとしての、あるオペレーションの遂行によって”生成”したオブジェクトである。換言すれば結果は、リクエスタのリクエストによってレスポндаによってあるオペレーションが遂行された直後にレスポндаからリクエスタに転送されるオブジェクトである。

【0445】サブクラス (subclass) : サブクラスは、1以上のクラスからの方法、特性及び又はフィーチャの定義を受け継ぐ。サブクラスは、他のクラスから受け継がなかった1以上のプロパティ、メソッド及び又はフィーチャを定義することが出来、そのフィーチャの新しいインプリメンテーションの定義によって、他のクラスから受け継がなかったフィーチャを再インプリメントすることができる。

【0446】スーパークラス (superclass) : あるサブクラスがそれからプロパティ、メソッド及び又はフィーチャを受け継いだクラスはそのサブクラスのスーパークラスである。

【0447】仮想的なプレイス (virtual place) : エンジンプレイスでないどんな場所も仮想的なプレイスである。

【0448】本発明の原理によれば、複数の相互接続されたコンピュータシステムに含まれる選定されたコンピュータシステムにある特別なコンピュータプロセスをルーティングし、その選定されたコンピュータシステムにおいて特定のコンピュータプロセスを実行するために用いられる。本発明のコンピュータプロセス (複数)

は、一実施例によれば、オブジェクト指向コンピュータプロセスとして定義される。本発明のコンピュータプロセスは、(i) プロセス中に、色々な操作を定義するインストラクションセットと (ii) コンピュータプロセスの遂行においてインストラクションセットを解釈し、中央処理ユニット (CPU) の作動を制御するエンジンを含む。

【0449】以下に、詳述するように、特定のコンピュータプロセスは、ネットワーク内の目的コンピュータシステムを特定化し、特別なコンピュータシステムがそこに移送されることを指示するインストラクションを実行することによって、コンピュータネットワークを通るそれ自身の運動すなわち転送を指示する。特別なコンピュータプロセスは、目的コンピュータシステム内において実行されている間に、ネットワークの他箇所では使用可能とならない情報にアクセスすることができる。特別なコンピュータシステムはこの情報にアクセスして、他のどのコンピュータシステムに移行すべきかを定めるためにその情報を使用することができる。以下に、詳述するように、本発明は、従来技術には見られない可動度レベル、拡張性及び一般性をコンピュータプロセスに供与する。

【0450】従来技術に典型的に見られるリモートプログラミングパラダイム (remote programming paradigms) において、コンピュータプロセスは、その自身の運動を差し向けず、目的コンピュータシステムにおいて実行されるようにソースコンピュータシステムから送られる。コンピュータプロセスがそれ自身の運動をネットワークを通して差し向けない場合には、コンピュータプロセスは、目的コンピュータシステムにおいてコンピュータプロセスによって得られる情報に基づいて移行すべき別の目的コンピュータシステムを選出することができない。すなわちこのコンピュータシステムは、第2の目的コンピュータシステムに送られる前に、得られた情報とともに、ソースコンピュータシステムに戻らなければならない。本発明のコンピュータプロセスは、ネットワークを通してそれ自身の運動を差し向けることができるので、本発明のコンピュータプロセスは、伝統的なリモートプログラミングパラダイムの実質的な延長を意味している。コンピュータネットワークを通してそれ自身の運動を差し向けるプロセスを提供する従来のリモートプログラミングシステムは、一般性と拡張性に欠けている。これらのシステムは、(i) 同質的なコンピュータネットワークに限定されるか、又は (ii) 異質的なネットワークにおいて移行するプロセスが現在プロセスを実行しているどんなコンピュータシステムにも、すなわち移行中のプロセスがそれに向かっているコンピュータシステムにも特定の適応するインストラクションを有することを要求するため、一般性に欠けている。本発明によればあるプロセスは異質的なコンピュータネットワ



ークを経て移動し、プロセスのインストラクションの実行は、そのプロセスが実行されている特別なコンピュータシステムすなわちプロセスがそれに向かって移行している特別なコンピュータシステムとは関わりはない。いかに、詳述するように、以下に開示されるコンピュータインストラクションは、異質的なネットワークのどのコンピュータシステムによっても一様に具体化される。すなわち本発明に従って形成されるプロセスは、異質的なコンピュータネットワークのどのコンピュータシステムに移行して、そのコンピュータシステムの性質に関する特定な情報もなしにそこで実行されることができる。

【0451】従来技術のリモートプログラミングシステムは、プロセスがデータオブジェクトの、ある1つのクラスを作り出して、プロセスが後にそれに向かって移動するコンピュータシステム内においてそのクラスのデータオブジェクトを利用することが出来ないため、典型的に拡張性に欠けている。多くの今日のコンピュータシステムは”オブジェクト指向”型である。用語集で既に規定したように、1つのオブジェクトは、(i) ある数のプロパティによって限定された内部的状態と、(ii) ある数のメソッドによって規定された内部的な挙動と、

(iii) ある数のフィーチャによって規定された外部的な挙動とを有する。同様のプロパティ、メソッド及びフィーチャを有するある数のオブジェクトのグループのこれらのプロパティ、メソッド及びフィーチャは、1つのクラスによって規定される。従来技術のシステムによれば、あるクラスを規定するプロセスは、そのプロセスが後に移行するコンピュータシステム内において実行されている間にそのクラスのオブジェクトを使用することはできない。移行するプロセスと共にクラス定義を移動させることは、クラス定義が標準化されていないことと、クラス定義が一般に非常に大きく複雑であることと、単一のプロセスが一般に多くのクラスのオブジェクトを使用することによって、従来技術のシステムによって実行できないか、又は実用的ではない。

【0452】クラス定義は、従来技術のシステムが典型的には可動のプロセスを考慮して設計されたものではない現存のプログラミング言語に基づいているため、”標準化”されていない。従ってこれらの現存のプログラミング言語の1つを用いて形成されたプロセス中のクラス定義は、典型的には、プロセスがその内部において作り出されたコンピュータシステムに特異の情報に依存している。これらの情報の例はクラス定義の一部の特定のメモリアドレスである。特別なコンピュータシステムに特異のこれらの情報に依存するため、従来技術のシステムのクラス定義をリモートコンピュータシステムに転送することが非常に困難になっている。

【0453】本発明によればクラスは、開示されたコンピュータインストラクションセットの一部であるクラスオブジェクトによって表される。本発明に従って形成さ

れたプロセスは、1つのコンピュータシステムから別のコンピュータシステムに移行している時のクラスオブジェクトを含む。従ってあるプロセスは第1のコンピュータシステムにおいてオブジェクトクラスを定義し、第2のコンピュータシステムに移行しそしてその第2のコンピュータシステムにおいてそのクラスのオブジェクトクラスを利用することができる。移動するプロセスによって利用される全てのクラスの転送に必要なとされない遅延及びコストは、プロセスがそれに向かって移動するコンピュータシステムにおいて定義されていないクラスのみを転送することによって除かれる。

【0454】また、本発明は、いくつかの目的コンピュータシステムにある1つのプロセスのいくつかのクローンを同時に転送をする際に、従来技術に効率性を供与する。以下に詳細に説明するように、プロセスの1つのクローンの移送経路が他のクローンの移送経路と合致している限り、1以上のクローンの生成を遅延させることによって、実質的な効率性が実現される。クローン及び発明のこの局面においては以下に詳細に説明する。

【0455】遠隔プログラミングシステムにおいて2つのプロセスは、第1のプロセスへのアクセスを第2のプロセスに与える第1のプロセスによって、情報を共有する。第1のプロセスは、第2のプロセスへのアクセスを同時に得ることなしに第1のプロセスへのアクセスを第2のプロセスに与える。この状況において第2のプロセスは、第2のプロセスの安全性(セキュリティ)を犠牲にすることなしに、第1のプロセスの実行に自由に干渉することができる。本発明によれば、第1のプロセスは、第2のプロセスへのアクセスを得ると同時に、第3のプロセスの共同及び関係間によって第1のプロセスへのアクセスを第2のプロセスに同時に与える。第2のプロセスが形態に従って第1のプロセスとのこのような交換を受け入れるか、又は拒絶することによって、別のセキュリティが与えられる。第3のプロセスは、(i) 第2のプロセスがこのような交換に同意すること、及び(ii) 各々のプロセスが他のプロセスにへのアクセスを同時に取得し、それによってどのプロセスも他のプロセスに同様の機会を与えることなしには他のプロセスの実行に干渉することが出来ないことを確実にする。

【0456】本発明の新規なコンピュータプロセスは、エージェントプロセスである。エージェントプロセスは、アペンディクスA(この開示の一部であり、参照によって全体として本明細書の一部とされる)に開示しているコンピュータインストラクションによって形成されたコンピュータプロセスである。エージェントプロセスは、本発明の別な新規のコンピュータプロセスすなわちエンジンによって各々解釈され、それによって、以下に、及びアペンディクスAに、開示されるコンピュータインストラクションを実行する。”解釈する”と言う用語はこの明細書においてはこの技術に理解されているとおりに

用いられてる。すなわち一連のコンピュータインストラクションの中の1つのコンピュータインストラクションがエンジンによって読まれそして実行された後に、一連のインストラクションの中で次のコンピュータインストラクションが読み出される。

【0457】ここに開示されたインストラクションセットのインストラクションをコンパイルするのではなく解釈することによって、より大きな一般性が供与される。第1のエージェントは第1のコンピュータシステムから第2のコンピュータシステムに移行し、そこで第1のエージェントにプロシーダを与える。第2のエージェントと会合することができる。第1のエージェントは次に第1のエージェントが本来実行するように設計されていなかったプロシーダを次に実行されることができる。

【0458】本発明の更に別の新規なコンピュータシステムは、プレイスプロセスである。プレイスプロセスはコンピュータネットワークを通じて分布されている。本来のエージェントプロセスは、それぞれのプレイスプロセスを”占有”している。すなわち、プレイスプロセスは、エージェントプロセスの内部状態の一部であり、従ってエージェントプロセスが実行されるためのコンテキストを与える。各々のエージェントプロセスは、第1のプレイスプロセスから、第2のプレイスプロセスへのエージェントプレイスへの移送を開始し制御することができる。”エージェント”及び”プレイス”は本明細書に使用されている限りにおいて、”エージェントプレイス”及び”プレイスプロセス”のそれぞれの略記である。

【0459】図6は、通信リンク102ABによって接続されたコンピュータシステム120A及び120Bを含むコンピュータネットワーク100を示している。コンピュータシステム120Aはプレイス220A及びエージェント150Aを実行する。エージェント150Aは図6に示すようにプレイス220Aを占有している。コンピュータシステム120Bはプレイス120Bを実行している。以下において、あるエージェント又はプレイスが特定のプレイスを占有しているという表現は、そのエージェント又はプレイスがその特定のプレイスを含むコンピュータシステム内において実行されているという表現も含むように解釈とされるものとする。

【0460】エージェント150Aは、コンピュータシステム120Aにインストラクションを送出する。エージェント150Aは、このインストラクションにตอบสนองして、プレイス例えばプレイス220B（インストラクション中において特定されている）に移送される。このインストラクションはオペレーション”go”と呼ばれ、エージェント150Aによるインストラクションの送出手は、ここではエージェント150Aによるオペレーション”go”の遂行と呼ばれる。エージェント150Aによってオペレーション”go”が遂行されると（i）エー

ジェント150Aの実行は中止され、（ii）エージェント150Aは、その実行状態を保存する標準化形態に符号化され、（iii）エージェント150Aの標準化形態はコンピュータシステム120Bに移送され、（iv）保存された実行状態を含むエージェント150Aは標準化形態から（デコード）され、（v）コンピュータシステム120B内においてエージェント150Aの実行が再開される。エージェント150Aによってオペレーション”go”が遂行されたのちに、エージェント150Aは、もはやプレイス220Aを占有してなく、コンピュータシステム120Aにおいて実行されていない。その代りにエージェント150Aは、プレイス220Bを占有し、コンピュータシステム120B（図7）内において実行されている。エージェントの実行の最中に遠隔のコンピュータシステムにエージェントが移行し得ることによって、そのエージェントはそれが、アクセスするように構成されているデータに向かって自由に移動することができる。

【0461】標準化された形式とは、エージェントが1つのコンピュータシステムから別のコンピュータシステムに転送されるようにエージェントが符号化される形態である。標準化形態は、符号化されたエージェントの実行状態を保存しているので、そのエージェントは、目的のコンピュータシステムにおいてデコードされることができ、エージェントの実行は、オペレーション”go”に順次後続するエージェントのインストラクションを実行することによって再開されることができる。本発明のネットワークの各々のコンピュータシステムは、エージェントに含まれるコンピュータインストラクションの実行に適したどんな形式においてもエージェントを保存することができる。しかし、1のコンピュータシステムから次のコンピュータシステムにエージェントを転送する場合に、ネットワークの各々のコンピュータシステムは転送の間あるエージェントの標準化形式を使用しなければならない。これは、単一の言語のシンタックス及び語彙に互いに交信するために同意しなければならないことと全く同様である。標準化形式の性質は、本明細書の一部分とされ、全体として参照によって本明細書に組み込まれるアペンディックスB、C、D、E及びFに一層詳細に説明されている。

【0462】あるエージェントはそれ自身の運動をネットワークを通じて差し向けることすなわち転送することができるので、第1のコンピュータシステムから第2のコンピュータシステムへのトリップ（旅行）のある数のパラメータを特定するための手段を設けなければならない。この旅行の第1のパラメータは、エージェントがそれに向かって移送されるプレイスすなわち旅行の目的地である。一実施例によれば、トリップの目的値は、ネーム、クラス又はアドレスによって指定することができる。アドレスは一例としてローカルエリアネットワーク

内のある番地（ロケーション）である。

【0463】2以上のバスウェイ（通路）が第1のコンピュータシステムと第2のコンピュータシステムの間に存在していることがしばしばある。一方のバスウェイは迅速で、高価、他のバスウェイは、時間を費やすが安価であることがある。このシステムにおいては、いかなる単一のバスウェイも、あるエージェントの転送において常に好ましくない。従ってトリップの第2のパラメータは、エージェントの移送のための“way”及び“means”である。

【0464】又、近くのコンピュータシステムへのトリップのためにエージェントを構成する場合のエラーによって、そのエージェントがエージェントの作成者にとって非常に大きなコストで、遠隔のコンピュータシステムに誤って転送されることがある。そのため、トリップの重要な一つのパラメータは、トリップがアポートされる前に、エージェントを転送するために費やされる時間又はリソースの量である。

【0465】一つのネットワークを通してエージェントを転送することを制御する上の別の重要な考慮事項は、セキュリティである。一例として、ある特別のコンピュータシステムに向かって移動している全てのエージェントがイングレス（Ingress）、すなわち入ることを許可された場合、その特別のコンピュータシステムは、実質的な処理を必要とする多くのプロセスによって過負荷された状態になり、それによって特別のコンピュータシステムのスループットが低下することがある。従って、トリップの別のパラメータは、目的プレイスにおいて必要とされるエージェントのリソースの量である。従って、エージェントによって要求されるリソースが、そのプレイスが供与する意志、能力又は意図を持つ以上である場合には、エージェントは、そのプレイスへのイングレスを拒絶されることがありうる。

【0466】一実施例によれば、チケット1306（図6）は、エージェント150Aの転送を制御し、目的プレイスとしてのプレイス220Bを特定する。図6に示す様に、エージェント150Aは、チケット1306を含む。チケット1306は、プレイス220Aからプレイス220Bへのエージェント150Aのトリップを規定する。一実施例によれば、チケット1306は、

（i）そのトリップの目的プレイス、（ii）目的プレイスに向かうためにエージェント150Aが取るべき“way”又はバスウェイ及び“means”、（iii）トリップがアポートされる前に、トリップを完成させている必要のある最大量の時間、及び（iv）目的プレイスにおいて、エージェント150Aが使用することを求めるリソースの量、を特定化する。プレイス220Bは、プレイス220Bにおいてエージェント150Aが使用することを許可されているリソースの量を特定することによって、プレイス220Bが供与する様に設計されてい

る、より多くのリソースを、エージェント150Aが必要とするか否かを定めることができる。この状態では、プレイス220Bは、エージェント150Aに対してイングレス（進入）を拒絶することができる。すなわち、以下に一層詳細に説明するチケットを使用することによって、エージェントは、第1のプレイスと第2のプレイスとの間の継続中のトリップを完全に規定することができる。

【0467】前述した様に、本発明者が知る従来技術によるプロセスは、以前に占有されていたコンピュータシステムにおいて規定されたクラスのオブジェクトしか操作することができない。換言すれば、従来技術によるプロセスがオブジェクトクラスを規定した後に、別の目的コンピュータシステムに移行した場合には、そのプロセスは、目的コンピュータシステムが、同一のクラスの定義を含むか、又はそのプロセスが目的コンピュータシステム内の同一クラスを規定している場合を除いて、そのクラスのオブジェクトを操作することができない。以下に詳細に述べるように、一つのプレイスから次のプレイスに移動する本発明のエージェントは、目的プレイスにおいては規定されていないクラスの定義を転送する。すなわち、本発明によるエージェントは、第1のコンピュータシステム内のクラスを定義し、第2のコンピュータシステムに移動し、第2のコンピュータシステムにおいて実行される間に、そのクラスのオブジェクトを操作することができる。従って本発明によるエージェントは、従来の知られたプロセスに比べて著しい改良を表している。

【0468】従来の技術において一般に欠如しているのは、その他に、コンピュータプロセスがその内部において移動するコンピュータネットワーク中の複雑でマルチレベルのセキュリティシステムを具体化するための手段である。本発明によって提供され、従来技術においては示唆されていない一つのセキュリティ形式は、プレイスによって提供されるセキュリティである。以下に詳述する様に、プレイスは、色々なエージェントへのイングレスを許可したり拒絶したりすることによって、色々なレベルのセキュリティを提供し、幾つかのプレイスは、マルチレベルの階層形セキュリティを提供するように構成することができる。

【0469】本発明の色々な局面及び実施例を理解しそして知覚化することを助けるために、色々な代表的な及び相対的なコンベンションが図面において用いられている。本説明中に用いられている例示的及び相関的なコンベンションは、図8、図9及び図10に明示されている。コンピュータシステム120A（図8）はプレイス220A、プレイス220X、プレイス220Y、エージェント150A、エージェント150X及びエージェント150Yを実行する。コンピュータシステム120Aは、又、オブジェクト140A、140B及び140

Xをメモリ例えばコンピュータシステムのマスメモリ又はメインメモリ（どちらも図示しない）中に備えている。エージェント150A、150Xは、プレイス220Aを占有し、プレイス220Yは、プレイス220Xを占有し、エージェント150Yはプレイス220Yを占有している。エージェント150Aは、オブジェクト140A及び140Bを所有し、エージェント150Xはオブジェクト140Xを所有している。占有及び所有の相互関係については、以下に一層詳細に説明する。簡単に説明すると、プレイスの占有は、セキュリティの色々なレベル及び形式を供与し、所有は、オペレーション”go”を実行するエージェントと共にどのオブジェクトが移動するかを定める。

【0470】図9は、図8に示され、かつ図8について説明された色々の相互関係の別途の、また同等の、表現形式を表している。図8は、エージェント150Xがプレイス220Aを占めること、及びオブジェクト140Xを収容していることを正確に表しているが、図8の形式は、より複雑な相互関係を表すには適してはいない。従って、図9のトリー構造は、本発明の色々な実施例の色々なコンピュータインストラクションを表す際に、より複雑な相互関係を表すために図面において用いられている。

【0471】図9のトリー構造は、図10に部分的に示したクラスハイエラキートリーと混同されるべきではない。図10は、図8、図9に示した色々のアイテムのクラス関係を表している。オブジェクト140A、140B及び140Xは、クラスオブジェクト520により表されないクラス”object”のメンバである。クラスのメンバシップは、前述の用語表に詳細に説明されており、クラスとそのクラスのメンバであるオブジェクトとの間の鎖線によって表されている。クラスオブジェクト522は、クラス”object”のサブクラスであるクラス”Process”を表している。プレイス220X、220Y及び220Aは、クラス”Place”のメンバであり、このクラス”Place”は、クラス”Process”のサブクラスであり、クラスオブジェクト524によって表されている。エージェント150A、150X及び150Yは、クラス”Agent”のメンバであり、このクラス”Agent”は、クラス”Process”のサブクラスであり、クラスオブジェクト526によって表されている。従ってプレイス220X、220Y及び220A及びエージェント150A、150X及び150Yは、全てクラス”Process”のメンバであり、従ってコンピュータプロセスである。オブジェクト140A、140B及び140Xは、クラス”Process”のメンバではなく、従ってコンピュータプロセスではない。

【0472】あるエージェントがあるプレイスから別のプレイスに転送される間に、エージェントによって所有されたオブジェクトは、一つのプレイスから次のプレイ

スへとエージェントと共に転送される。しかし、あるオブジェクトの転送は、そのオブジェクトが大きい場合に相当多くのリソースを消費する。一般に、第1のコンピュータシステムにおいて実行されている第1のプレイスから第2のコンピュータシステムにおいて実行されている第2のプレイスにオブジェクトを含むエージェントを転送するための時間は、既に第2のプレイスに存在しているオブジェクトの転送を除去することによって実質的に減少する。これは簡単な例を考察することによって容易に示される。エージェント150A（図11）は、コンピュータシステム120Aにおいて実行されており、やはりコンピュータシステム120Aにおいて実行されているプレイス220Aを占有している。エージェント150Aはオブジェクト140A、140B及び140Cを所有している。この実施例では、オブジェクト140Bはダイジェスト622を有する。ダイジェスト622は、オブジェクト140Bが相互変換可能であり、ダイジェストは622に等しいダイジェストをもついかなるオブジェクトもオブジェクト140Bに代えることができることを示している。

【0473】プレイス220Bは、オブジェクト624及び626を所有し、従って収容している。オブジェクト624はダイジェスト628を有する。あるプロセスは、そのプロセスによって所有される全てのオブジェクト、及び、クラス（プロセス及びそのプロセスによって所有されるオブジェクトをメンバとする）を収容している。

【0474】前述した様にオペレーション”go”を遂行する場合に、エージェント150A及びエージェント150Aによって収容された全てのオブジェクトは、標準化形式において表わされる。しかしオブジェクト140Bは、エージェント150Aの標準化形式には含まれない。その代わりに、ダイジェスト622のコピーすなわちダイジェスト622-C（図12）がエージェント150Aの標準化形式に含まれている。エージェント150A、オブジェクト140A及び140C、ダイジェスト622-Cは、矢印Aの方向に通信リンク102ABを経てコンピュータシステム120Bに転送される。オブジェクト140Bは、少なくともコンピュータシステム120B内において同等の相互変換可能なオブジェクトが見出されたことが定められるまでは、コンピュータシステム120A内に保持されている。

【0475】コンピュータシステム120B（図13）、エージェント150A、オブジェクト140A、140C及びダイジェスト622-Cは、デコードされる。エージェント150Aをデコードする際に、コンピュータシステム120Bは、ダイジェスト622-Cを認識し、同等のダイジェストを持ったオブジェクトがプレイス220Bを占有しているかどうかを定める。プレイス220Bを占有するオブジェクト624は、ダイジ

エスト622に等しく、従ってダイジェスト622-Cに等しいダイジェスト628を有する。ダイジェスト628はダイジェスト622に等しいので、オブジェクト140B、624は相互変換可能である。従って、エージェント150Aをデコードする際に、コピー、例えばオブジェクト624-Cが、オブジェクト624について作成され、エージェント150A内において相互変換可能なオブジェクト140Bに代えられる。従ってエージェント150Aは、オブジェクト140Bの代わりに、オブジェクト624のコピーであるオブジェクト624-Cを所有する。従ってプレイス220Bへのオブジェクト140Bの転送は割愛される。同等の相互変換可能なオブジェクトがコンピュータシステム120B内に見出されるので、コンピュータシステム120Aからオブジェクト140Bを削除することができる。勿論、相互変換可能な同様なオブジェクトがコンピュータシステム140Bに見出されない場合もある。このような場合にはオブジェクト140B(図12)は以下に詳細に説明するようにしてコンピュータ120Aから検索される。

【0476】第1のプレイスを占有するエージェントは、エージェントの1以上のクローンプロセスを作成して、各々のクローンプロセスをそれぞれのプレイスに転送することができる。従って実際には、あるエージェントは、幾つかのプレイスに移動して、そのプレイスを同時に占有することができる。勿論同時に移動するのは単一のエージェントではなく、そのエージェントの複数のクローンである。

【0477】あるエージェントは、幾つかのプレイスに同時に移行し占有するために、あるインストラクションを送出する。このインストラクションは、エージェントの複数のクローンを作成し、各々のクローンをそれぞれのプレイスに移行させるインストラクションである。各々のクローンは一つのエージェントであり、クローンの作成の際において、クローンは、元のエージェントと同一であるため、元のエージェントのものと同一の実行状態を備えている。

【0478】一例として、コンピュータ120Aのプレイス220Aを占めるエージェント150A(図14)は、一以上のプレイスを占有するために、エージェント150Aのクローンを作成して、そのクローンを転送するインストラクションを、コンピュータシステム120Aに送出する。これらのプレイスは、例えば、送出されたインストラクションに特定化されている、コンピュータシステム120Bのプレイス220B及びコンピュータシステム120Cのプレイス220Cである。このインストラクションはオペレーション"send"と呼ばれ、エージェント150Aによるこのインストラクションの送出は、ここでは、エージェント150Aによるオペレーション"send"の遂行と呼ばれる。図14、図15、

図16及び図17に示したエージェント150Aによるオペレーション"send"の遂行は、この実施例では、エージェント150A内の二つのチケット(図示しない)によって管理される。これらの二つのチケットは、それぞれのクローンの転送を制御すると共に、エージェント150Aのそれぞれのクローンが向けられるプレイスとしてのプレイス220Bと220Cを特定化する。

【0479】エージェント150A-1、150A-2(図15)は、エージェント150Aから形成されたクローンであり、エージェント150A-1、150A-2がプレイス220Aを占有しないことを除いては、エージェント150Aと同一である。前述したように、エージェント150A-1、150A-2の実行状態も、最初は、エージェント150Aの実行状態と同一である。エージェント150A-1、150A-2は、以下に詳細に説明する様に、インターコンピュータ通信リンク120ABC(図16)に沿って、それぞれのプレイス220B、220Cに向かって移動する。

【0480】エージェント150Aによるオペレーション"send"の実行後(図17)に、エージェント150Aは、プレイス220Aの占有を継続する。エージェント150A-1は、プレイス220Bを占有し、エージェント150A-2は、プレイス220Cを占有する。

【0481】コンピュータシステム120A中のスペースとエージェント150Aのクローンの転送に要する時間とは、あるクローンの走行経路が別のクローンの走行経路と合致する限り、すなわち二つのクローンが共延の走行初部分を多少有する限り、エージェント150Aの完全なクローニングを遅らせることによって節減される。例えば図18において、エージェント150Aの両方のクローンは、それぞれの目的プレイス220B、220Cに到達するためにコンピュータシステム120Dを通過しなければならない。従ってエージェント150Aの単一のクローン、すなわちエージェント150A-1のみが、コンピュータシステム120Dに転送される。エージェント150Aの第2のクローン、すなわちエージェント150A-2はエージェント150A-1からコンピュータシステム120Dにおいて作成される。エージェント150A-1、150A-2は、次にそれぞれの目的プレイス220B、220Cに転送される。従って、エージェント150Aの単一のクローンのみがコンピュータシステム120Aによって形成され、単一のクローンのみがコンピュータシステム120Dに転送されるので、コンピュータシステム120Aのスペースが節減されると共に、エージェント150A-1、150A-2をそれぞれの目的地に転送するための時間も節減される。

【0482】二つのエージェントは、これらの二つのエージェントの間のミーティングに参加することによって情報を交換する。このミーティングにおいて、各々のエ

ージェントには、他のエージェントに対する参照が供与される。以下に説明する様に、第2のエージェントへの参照を有する第1のエージェントは、(i) 第2のエージェントに含まれる指示に従って特定の措置を取る様に第2のエージェントに指示し、(ii) 第2のエージェントに対してデータを送受信することができる。本発明の一局面によれば、エージェントは、第2のエージェントが最初に述べたエージェントへの参照を与えられていない限り、第2のエージェントへの参照をいかなるエージェントにも与えないことによって、他のエージェントとの干渉が防止される様にする。これにより、相互アクセスを許容することなしに、いかなるエージェントも第2のエージェントにアクセスすることができなくなる。

【0483】あるプレイスを占有する第1のエージェントは、そのプレイスを占有する第2のエージェントとのミーティングを開始することができる。このミーティングにおいて、第1のエージェントは、第2のエージェントにオブジェクトを転送したり第2のエージェントからオブジェクトを受け取ったりすることができ、また第2のエージェントは、第1のエージェントにオブジェクトを転送したり、第1のエージェントからオブジェクトを受け取ったりすることができる。

【0484】クラス"Place" (図10) のサブクラスは、"ミーティングプレイス" クラス (図示しない) である。図19～図24の例示的な実施例によれば、プレイス220Bは、クラス"ミーティングプレイス" の1つのメンバであるため、ミーティングプレイスである。したがってプレイス220Bは、この明細書において時に"ミーティングプレイス220B" と称される。コンピュータシステム120Bにおいて実行されているミーティングプレイス220B (図19) は、ミーティングプレイス220Bを占有しているエージェント150A、150Bに、矢印A、Bによって示すように1つのミーティングにおいて情報を交換し分け合うための手段を提供する。エージェント150A中のペティション3106は、エージェント150A、150Bの間のミーティングを規定し管理する。

【0485】エージェント150A、150Bの間のミーティングは、図20～図24に示すように構成される。エージェント150A (図20) は、エージェント150A、150Bの間のミーティングを準備するようにミーティングプレイス220Bに指示するための、矢印851によって表された第1のコンピュータインストラクションを送出する。第1のインストラクションは、ミーティングを管理し、エージェント150Aがエージェント150Bとミーティングすることを望んでいることを特定するペティション3106を含む。エンジンは、矢印852によって示された第2のコンピュータ指令をミーティングプレイス220B (図21) のために発生する。このインストラクションは、エージェント1

50Aが第1のコンピュータインストラクションを送出したことをエージェント150Bに通知するとともに、エージェント150A、150Bの間のミーティングが承認可能か否かを定めるためのある数のコンピュータ指令をエージェント150B中において実行させるインストラクションである。

【0486】エージェント150A、150Bの間のミーティングが、承認可能であることが定められたら (図22)、エージェント150Bは、矢印853によって表される回答を第2のコンピュータ指令に対して送出する。これによってエンジンは、エージェント150A、150Bの間のミーティングを準備するようにミーティングプレイス220Bのためにエンジンに指示する。エンジンは、ミーティングプレイス220B (図23) のために、エージェント150Bへの参照をエージェント150Aに送出する (この送出は矢印854Aによって示される)。またエンジンはエージェント150Aへの参照を150Bに送出する (この送出は矢印854Bによって示される)。エージェント150A (図24) は、エージェント150Bへの参照を用いて、エージェント150Bと相互作用する。この参照は、矢印855Aによって示される。さらにエージェント150Bは、エージェント150Aへの参照を用いて、エージェント150Aと相互作用を行う。この参照は矢印855Bによって示される。

【0487】2つのエージェントは、これら2つのエージェントは同一のプレイスを占めない限りミーティングにおいて一緒に参加することはできない。プレイスのハイエラキーは、可変のレベル及び形式のアクセス及びセキュリティをいろいろなエージェントに与えるためのメカニズムを本発明のユーザに供与する。アクセス及びセキュリティのいろいろなレベル及び形式の利点は、図25～図29の例示的な実施例によって示される。

【0488】図25は、フロア902-1、902-2、902-3、902-4、902-5を有する建物902を示している。フロア902-3は建物902のプレイス内の1つのプレイスでありしたがって建物902のサブプレイスである。図26はルーム、902-3-1、902-3-2、902-3-3、902-3-4を有するフロア902-3を示している。ルーム902-3-2はフロア902-3のプレイス内のプレイスでありしたがってフロア902-3のサブプレイスである。

【0489】建物902の組織はハイエラキー的なセキュリティにとって有利である。一例として建物902は、第1のセキュリティレベルの人々たとえば陸軍関係者のみが入ることを許可されるように制限されることができる。フロア902-3は、第2のレベル (より高いレベル) のセキュリティレベルの人々たとえば海軍関係者のみが入り得るようにさらに制限することができる。

ルーム902-3-2は、第3の（より高いレベル）セキュリティレベルの人々たとえば海軍将校のみが入り得るようにさらに制限されることができる。フロア902-4は、フロア902-3と無関係にたとえば空軍関係者のみが入り得るようにさらに制限することができる。

【0490】コンピュータシステム120X（図27）中のエンジンは、プレイス220X1、220X2を実行する。プレイス220X1は、一例として建物902を表すことができる。プレイス220X1-1、プレイス220X1-2（一例としてフロア902-4、902-3を表す）は、プレイス220X1（図28）を占有し、したがってプレイス220X1のサブプレイスである。逆にプレイス220X1はプレイス220X1-1、220X1-2のサブプレイスである。たとえばルーム902-3-1を表すプレイス220X1-2-1は、プレイス220X1-2（図29）を占有する。プレイス220X1-2-1はしたがってプレイス220X1-2のサブプレイスでありプレイス220X1-2はプレイス220X1-2-1のスーパープレイスである。

【0491】本発明の一実施例によれば、2つのエージェントは、“対面のみ”ですなわち2つのエージェントが同一のプレイスを占める場合にのみミーティングすることができる。ここで本明細書で使用される限り、あるエージェントはただ1つのプレイスのみを占め、そのプレイスのサブプレイス又はスーパープレイスを同時に占有することはない。一例としてプレイス220X1-2を占有するエージェントはプレイス220X1又はプレイス220X1-2-1を同時に占有することはない。建物902（図25）にいる1番目の人が、その人がフロア902-3にもいない限りフロア902-3の2番目の人と対面で交信することができないのと同様に、プレイス220X1（図28）を占有するエージェントは、プレイス220X1-2を占有するエージェントとは交信できない。フロア902-3（図26）にいる1番目の人は、ルーム902-3-2にもいない限り、ルーム902-3-2にいる2番目の人と対面で交信できないのと同様に、プレイス220X1-2（図29）を占有するエージェントは、プレイス220X1-2-1を占有するエージェントとは交信できない。

【0492】このプレイスハイエラキーは、本発明のユーザがプレイス220X1へのアクセスを制限し、プレイス220X1-2へのアクセスをさらに制限し、プレイス220X1-2-1へのアクセスをさらに制限することを可能にする。さらにここに具体化されたセキュリティハイエラキーは、具体化されたプレイスハイエラキーに必ずしも直接に関連しているのではない。一例としてプレイス220X1-2へのアクセスは、プレイス220X1-2のサブプレイスたとえばプレイス220X1-2-1へのアクセスよりもさらに制限的とすること

ができる。あるプレイスへのアクセスの制限は以下に詳細に説明されている。このようにプレイスハイエラキーは、いろいろなエージェントにあるプレイス、これらのプレイスのサブプレイス及びこれらのサブプレイスのスーパープレイスを占めている他のエージェントへのいろいろなアクセスレベルが与えられるような複雑なセキュリティハイエラキーを本発明のユーザが構成することを可能にする。

【0493】以上に説明した新規なプロセスの詳細をさらに記述するためには、以上に説明したコンピュータシステムの構造のよりよき理解が必要とされる。図30に示した実施例において、三つのコンピュータシステムはコンピュータネットワーク100を形成するように接続されている。コンピュータシステム120Aは、CPU110A、ネットワーク通信ハードウェア104A及びメモリ117Aを含む。図1の入力モジュール12及び出力モジュール14に対応する入力モジュール及び出力モジュールはわかりやすさのために割愛されている。さらに、図1のマスメモリ17A及びメインメモリ17Bは、メモリ117Aを形成するように組み合わせられている。ネットワーク通信ハードウェア104Aは、CPU110Aがネットワークを横断して信号を伝搬させたりネットワークから信号を受けて解釈したりすることを可能にするどんな装置でもよい。

【0494】メモリ117Aには、CPU110A内において同時に動作するある数のコンピュータプロセスが存在する。ネットワークマネージャ130Aは、コンピュータネットワークのいろいろのコンピュータシステム間のデータ伝送を連係化するコンピュータプロセスである。オペレーティングシステム131Aは、いろいろのコンポーネントの動作及びコンピュータシステム120Aのリソースを連係化するコンピュータプロセスである。一例としてオペレーティングシステム131Aは、CPU110A、メモリ117A及び入力/出力モジュール（図30には示さない、図1参照）使用を連係化する。エンジン132Aは、本発明のオブジェクト指向コンピュータインストラクションセットのコンピュータインストラクションを解釈し、そのオブジェクト指向コンピュータインストラクションセット中に規定されたオブジェクトの形の情報を処理するコンピュータプロセスである。一例としてエンジン132Aは、プレイス220A（図6）とエージェント150A（これら2つは、本発明のオブジェクト指向コンピュータインストラクションセットから構成されたプロセスである）の同時実行を行う。エンジン132Aは、図6には示されていない。

【0495】コンピュータシステム120Aのメモリ117A（図30）内において実行されるこれらのプロセスの他に、典型的には1以上のプロセス（たとえばユーザアプリケーション133Aである）が存在する。ユー



ザアプリケーション133Aは、エンジン132Aによって解釈されるオブジェクト指向インストラクションセット内において規定されたオブジェクトの生成及び／又はマニピュレーションをエンジン132Aに要求することができる。

【0496】コンピュータシステム120B、120Cは、コンピュータシステム120Aと同様に構成されている。しかし、コンピュータシステム120A、120B、120Cの一般的な構造は同様であるが、コンピュータシステム120A、120B、120Cはその他の点では異質的である。

【0497】コンピュータネットワーク100のコンピュータシステムは、1つのコンピュータシステムから別のコンピュータシステムにエージェントプロセスが転送され得るように接続されている。コンピュータシステム120A、120B、120Cは、通信リンク102AB、102BC、102ACによってそれぞれのネットワーク通信ハードウェア104A、104B、104Cを結合することによってコンピュータネットワークを形成するように接続されている。通信リンク102AB、102BC、102ACは、1つのネットワーク通信ハードウェアたとえばネットワーク通信ハードウェア104Aから別のネットワーク通信ハードウェア、たとえばネットワーク通信ハードウェア104Bにデータを転送することができるどんな手段でもよい。一例として、ネットワーク通信リンク102ABは、公共に対して交換される電話ネットワークでよい。この場合ネットワーク通信ハードウェア104A、104Bはモデムであり、ネットワークマネージャ130A、130Bは、通信リンク102ABを介した通信すなわち公共に対して交換される電話ネットワークを設立し利用するために、ネットワーク通信ハードウェア104A、104Bにインストラクションを送出することができる。

【0498】オブジェクト指向コンピュータインストラクションセットから成るオブジェクトは、エンジンたとえばエンジン132A（図31）によって実行される。エンジン132Aは、通信下部構造132A-CI、プログラム部分132A-P及びデータ部分132A-Dを有する。エンジン132Aによって実行されるオブジェクト、たとえばオブジェクト140Aの状態を表すデータは、エンジン132Aのデータ部分132A-Dに記憶されている。エンジン132Aのデータ部分132A-Dは、エンジン132Aのワークスペースとして保留されたメモリ117Aのメモリスペース（図30）であり、コンピュータシステム120A上の他のプロセスたとえばオペレーティングシステム131A及びユーザアプリケーション133Aの見地から一般にはアクセスすることはできない。

【0499】前述したようにエンジンはプロセス及びオブジェクトを実行する。エンジン132Aのプログラム

部分132A-P（図31）は、データ部分132A-Dにおいて表されるオブジェクトを実行する。プログラム部分132A-Pを形成するように組み合わせられるコンピュータインストラクションは既知のコンピュータ言語のものとして行うことができる。一例としてマイクロフィルムアペンディクスGとして添付したコンピュータソフトウェアは、本発明の原理にしたがって構成されたエンジンのプログラム部分であり、C++プログラミング言語にしたがって構成されている。

【0500】エンジン132Aの通信下部構造132A-CIは、ネットワーク100を通じて分散されたエンジンの間にデータを転送するためのコンピュータインストラクションを含む。

【0501】マイクロフィルムアペンディクスHとして添付されたコンピュータソフトウェア（この明細書の一部であり、全体として引照されることによって本明細書の一部となる）は、本発明の原理にしたがって構成されたエンジンの通信下部構造である。エンジンの通信下部構造の多くの様相は、この明細書の一部となり、引照によって全体として本明細書に組み込まれるアペンディクスDに一層詳細に説明されている。

【0502】図32は、図31の、コンピュータネットワークの代替となる同等の表現形態である。

【0503】図31に示すように、コンピュータシステム120Bのエンジン132Bは、エンジン132Aの形態と直接類似する仕方で構成されている。

#### 【0504】ネットワーク中のオブジェクト

以下に一層詳細に説明するように、プレイス220A、220B（図6）、エージェント150A、チケット1306などは、本発明の原理にしたがって“オブジェクト”を用いて規定されている。前述したようにオブジェクト（エンジン132A、132Bによって解釈されるコンピュータインストラクションセットにしたがって形成され、エンジン132Aによって解釈される）は、データ部分132A-D（図31）中に記憶されている。一例として、データ132A-D中のオブジェクト140Aは、エンジン132A、132Bによって解釈されるコンピュータインストラクションセットにしたがって形成され、エンジン132Aによって解釈される。

#### 【0505】本発明のコンピュータインストラクションセットの概略

本発明の新規なプロセスの各々及びこれらのプロセスを限定し指示するために必要ないろいろなフィーチャは、本発明のエンジンによって解釈されるコンピュータインストラクションのセットによって限定される。本発明のコンピュータインストラクションは、オブジェクト指向である。したがって本発明のデータたとえばエージェント150Aを表すデータは、各々内部状態と外部挙動とを有するオブジェクトに組織化される。あるオブジェクトのプロパティは、そのオブジェクトの内部状態を規定

し、そのオブジェクトのフィーチャは、そのオブジェクトの外部挙動を規定する（前記の用語表参照）。各々のオブジェクトはある数のクラスの内の1つのクラスの1つの例（インスタンス）である。

【0506】本発明の原理によれば、以下に記述されるミックスインクラスを除いたコンピュータインストラクションセットに規定されたすべてのクラスは、アベンディクスAに一層詳細に説明されるクラス“Object”のサブクラスである。したがって、ここに記述されるミックスインクラスでない各々のクラスは、クラス“Object”のフィーチャ及びプロパティを受け継いでいる。

【0507】制限された多重の引継ぎは、一実施例によれば、ミックスインクラスを用いて具体化される。“Mix-ins”又は“Mix-in classes”は、クラス“Object”のサブクラスでないクラスである。以下に記述されたアベンディクスAにも記述されるミックスインクラスの例は、ミックスインクラス“Executed”、“Named”及び“Referenced”を含む。“flavor”又は“flavor class”とも呼ばれる非ミックスインクラスは、多くとも一つのフレーバのイミディエートサブクラスで有り得るが、0又は1以上のミックスインクラスのイミディエートサブクラスで有り得る。ミックスインクラスは、ノークラス又は別のミックスインクラスのイミディエートサブクラスで有り得る。他に指定しない限り、クラスはフレーバである。クラスハイエラキーにおいてサイクルは許容されないすなわち別のクラスのサブクラスであり又スーパークラスでもあることはクラスには許されない。

【0508】\*\*59頁\*\*

ミックスインクラスを使用すると、フィーチャ及びプロパティが一度限定されそして広い範囲のクラスにわたって使用される。例えばミックスインクラス“Ordered”は、二つのオブジェクトの相対的な順序を定める為のオペレーションを規定する。フレーバクラス“Association”、“Citation”、“List”、“Identify”、“Pattern”、“Permit”、“Bit”、“Boolean”、“Character”、“Number”、“Octet”及び“Time”は、ミックスインクラス“Ordered”から引き継がれる。従って関連、サイテーション、リスト、識別子、パターン、許可、ビット、ブーリアン、キャラクタ、数、オクテット、及び時間は同一のクラスの他のメンバに対してある順序を持っている。一例として数値の場合、数2は、数1の“後”であり数3の“前”である。

【0509】本発明においてオブジェクトの主要なクラスは、プロセスのクラスである。前述したようにプロセスは、(i)他のプロセスによって占有されるプレイス、又は(ii)(a)第1のプレイスからそれ自身を転送し第1のプレイスの占有を終了し第2のプレイスを占有することのできるエージェント及び(b)同一のプレイスを占有する他のエージェントと相互作用をし得る

エージェント、のどちらかである。

【0510】エージェント150A（図31）は、エンジン132によって実行される本発明のコンピュータインストラクションセットに従って形成されたプロセスである。エージェント150Aは、(i)それ自身を検査して変形し、(ii)ネットワーク100中の第1のプレイス（図30）から第2のプレイスにそれ自身を転送し、(iii)第2のプレイスを占有する他のエージェントと相互作用することができる。

【0511】本明細書においてプロセスは、アーギュメントを消費し0又は1つの結果を発生させるオペレーションを行なうものとして記述される。しかしプロセスそれ自体は、以下に一層詳細に説明するインストラクションの中から選定されたインストラクションの集合を含むオブジェクトであり、このインストラクションはCPU110A内において動作しているエンジン例えばエンジン132Aによって解釈される。あるプロセスによる動作の遂行又は開示されたインストラクションセットの他のオブジェクトの遂行は、実際には、プロセス又はオブジェクトに含まれる特別のインストラクション群の選定及びその解釈である。エンジンによるプロセスのインストラクションの解釈は、ここでは、“プロセスの解釈”とも呼ばれる。すなわち、エージェント150Aがあるオペレーションを実行する場合、エージェント150Aは、エンジン132Aにインストラクションを供給し、エンジン132Aは、そのインストラクションを解釈し、エージェント150Aによるオペレーションの遂行を行なう適当なタスクを実行するようにCPU110Aに指示する。プロセスとエンジンとの間の相互作用は以下に一層詳細に説明する。

【0512】本発明の重要な様相は、以下に一層詳細に説明するコンピュータインストラクションセットがネットワーク100（図30）のそれぞれのコンピュータシステムによって一様に具体化されていることである。コンピュータシステム120A、120B、120Cは、完全に異なって且つ整合性のないデータ構造メモリ形態CPU及びオペレーティングシステムを使用することができる。しかしエージェントはコンピュータシステム120A、120B、120Cのどれに対してもそれ自身を転送することができるので、各々のコンピュータシステムが標準化された形態で本発明のコンピュータインストラクションセットを具体化していることが大切である。コンピュータインストラクションが一様に具体化されている限り、コンピュータシステム120A120B、120Cが他の点で不整合であることは有り得る。従ってエージェントは均質なコンピュータネットワークだけでなく異質なコンピュータネットワークのコンピュータシステムの間にも自由に移動することができる。

【0513】典型的には、従来技術のシステムは、プロセスが均質なネットワーク内を移動すること、又はプロ

セスが実行中のコンピュータシステムと整合性のあるコンピュータインストラクションを特異的に実行するようにプロセスが構成されていることを必要としていた。後者のケースにおいては、プロセスは、そのプロセスの実行が開始されたコンピュータシステムの特別のシステム要求及びそのプロセスが移行しようとしている何れかのコンピュータシステムの特別のシステム要求に対して構成されていた。従って本発明のエージェントのコンピュータインストラクションは、異質なネットワークのいかなるコンピュータシステムに対しても実行され得るので、本発明は従来技術に対する実質的な改良を表している。

#### 【0514】エージェント

前述したようにエージェントの挙動は部分的にはそのエージェントの内部状態に依存する。従ってネットワーク中のあるエージェント外部挙動を検討する前に、エージェントの内部状態のいくつかの様相について簡単に説明する。各々のエージェントはクラス"Agent"のメンバである。

【0515】クラス"Agent"は抽象的であるから、いかなるエージェントもクラス"Agent"の例(インスタンス)ではないことに注意すべきである。クラス"Agent"は、オペレーション"live"のいかなるインプリメンテーションもクラス"Agent"によって規定されたり受け継がれたりしていないので、抽象的である。以下にまたアペンディクスAに一層詳細に説明するように、オペレーション"live"は、プロセスの生成時にそのプロセスすなわちエージェント又はプレイスによって実行されるステップを規定する。これらのステップは集散的にプロセスの"中央プロシージャ"と呼ばれる。エージェント及びプレイスが各々のユーザの特定のニーズに適合する中心プロシージャを本発明のユーザが設計し供与するので、エージェント及びプレイスの為の中心プロシージャは提供されない。従って、本発明のユーザは、クラス"Agent"の具体的なサブクラスを生成させ、オペレーション"live"の為のインプリメンテーションを供給する。

【0516】クラス"Agent"は、クラス"Process" (図5)のサブクラスである。クラス"Process"は、クラス"Object"のサブクラスであり、ミックスインクラス"Named"からも受け継いでいる。クラス"Object"は、ミックスインクラス"Referencd"からも受け継いでいる。あるエージェントは次の属性を所有している。これらの属性は(i)スーパークラス"Object"から受け継がれたアトリビュート(属性)"class"及び"size"、(ii)ミックスインクラス"Referencd"から受け継がれたアトリビュート"isProtected"、(iii)ミックスインクラス"Named"から受け継がれたアトリビュート"name"、及び(iv)スーパークラス"Process"から受け継がれたアトリビュート"bra

nd"、"Permit"及び"privateClasses"である。クラス"Agent"はいかなる属性も規定しない。前述した各々の属性及びクラスは、アペンディクスAにおいて一層詳細に説明される。

【0517】ダイアグラム1270 (図33)は、エージェント150Aがメンバであるクラス(複数)のクラス関係を表している。エージェント150Aは、クラス"Agent"を表すドメイン1272に含まれるように図示されているので、クラス"Agent"のメンバである。

【0518】クラス"Agent"は抽象的であるから、エージェント150Aは、クラス"Agent"の1以上のサブクラス(図示しない)のメンバでもある。

【0519】ドメイン1272は、クラス"Process"を表すドメイン1274中に完全に収容されている。従ってエージェント150Aは、クラス"Process"によって規定されたアトリビュート"brand"、"Permit"及び"privateClasses"を受け継ぐ。さらにクラス"Agent"のすべてのメンバは、クラス"Process"のメンバでもある。従って前述したようにクラス"Agent"はクラス"Process"のサブクラスである。

【0520】ドメイン1274は、クラス"Object"を表すドメイン1276中に完全に収容されている。従ってエージェント150Aは、アトリビュート"class"及び"size" (クラス"Object"によって規定される)を受け継いでいる。さらに、クラス"Agent"のメンバを含むクラス"Process"のすべてのメンバは、クラス"Object"のメンバでもある。従って前述したように、クラス"Process"は、クラス"Object"のサブクラスである。

【0521】クラス"Process"を表すドメイン1274は、クラス"Named"を表すドメイン1278中に収容されている。コネクション1278Aは、ドメイン1274がドメイン1278に収容されていることを示すと共に、クラス"Object"を表すドメイン1276がドメイン1278中には収容されてなく、ドメイン1278がドメイン1276に収容されていないことを明瞭に表している。

【0522】ドメイン1278によって表されたクラス"Named"はミックスインクラスである。ドメイン1274はドメイン1278に含まれているので、エージェント150Aは、ドメイン1278に含まれ従ってミックスインクラス"Named"のメンバである。ミックスインクラス"Named"は、エージェント150Aに含まれないアトリビュート"name"を規定している。

【0523】ミックスインクラス"Referencd"を表すドメイン1280は、ドメイン1276を含み、このドメイン1276は、コネクション1280Aによって示すように、クラス"Object"を表している。ドメイン1276はドメイン1280に含まれるので、エージェン

ト150Aは、ドメイン1280に含まれ、従ってミックスインクラス"Referencd"のメンバである。ミックスインクラス"Referencd"は、エージェント150Aに含まれるアトリビュート"isProtected"を規定する。

【0524】本発明の開示された実施例の各々のフレバは、クラス"Object"のサブクラスであるので、各々のフレバはミックスインクラス"Referencd"のサブクラスでもある。従って、クラス"Object"の特徴からクラス"Referencd"の特徴を分離する必要はない。しかしミックスインクラス"Referencd"の特徴は、本発明の概念形成及び理解を助ける為に、クラス"Object"の特徴から分離されている。

【0525】用語表において述べたように、オブジェクトは、参照によって設定された開示されたコンピュータインストラクションの範囲内において特定化される。以下に詳述するように、オブジェクトにある操作を行なうように指示する場合、オブジェクトは、参照によって特定化される。ミックスインクラス"Referencd"によって定義された特徴は、オブジェクトを特定化する参照に基づいて動作する。一例としてアトリビュート"isProtected"は、オブジェクト自身でなくオブジェクトへの参照が保護されるか否かを定める。クラス"Object"によって定義された特徴は、参照によって特定化されたオブジェクトに対して動作する。一例としてアトリビュート"class"はそのオブジェクトがどのクラスの例(インスタンス)であるかを定める。

【0526】エージェント150A(図33)によって占有されたプレイスに関する情報を供与する属性が定義されていないことに注意されたい。しかしエージェント150Aによって占有されたプレイスは、エージェント150Aの一つの特性として保存される。エージェント150Aは、"here(ヒア)"セレクトの実行によってどのプレイスをエージェント150Aが占有するかを定めることができる。セレクト特に"ヒア"セレクトの実行は、アペンディクスAに詳述されている。エージェント150Aによって占有されているプレイスである特性は、クラス"Process"によって定義される。従っていかなるプロセスも、すなわちプレイス又はエージェントは、プロセスによって占有されているプレイスである特性を含んでいる。

【0527】ミックスインクラス"Named"のイミューティエートサブクラスである本明細書及びアペンディクスAに開示されたクラスのみがクラス"Process"であるが、クラス"Process"のサブクラスでないミックスインクラス"Named"のサブクラス(複数)は、本明細書及びアペンディクスAに開示されたインストラクションを用いて本発明のユーザによって定義される。従ってクラス"Named"はミックスインクラスである。

【0528】プロセスとしてのエージェント

各々のプロセス(エージェント及びプロセス)には、前述したプロセスの一義的な挙動を規定する中心プロシージャが組み合わされている。あるプロセスの中心プロシージャは、プロセスによって遂行されるオペレーション"live"を具体化する方法である。エンジンはそのプロセスにオペレーション"live"を遂行させ、それによりプロセスの中心プロシージャを実行させることによってプロセスの処理を開始する。プロセスの方法の供給については以下に詳述する。あるプロセスがオペレーション"live"の実行を成功のうちに、又は他の形で終了させた場合、プロセスは終了する。プロセスの終了はアペンディクスAのセクション2.4.11に説明されており、この説明は引用によって本明細書の一部とされる。

【0529】エージェントの可動度オペレーション"go"

前述したように、オブジェクトはオペレーション"go"の遂行によって1つのプレイスから次のプレイスへと移動する。オペレーション"go"は図34、図41、図50、図51、図53の例示的な例のコンテキストにおいて説明される。プレイス220Aからプレイス220Bに移動する為に、エージェント150A(図34)は、オペレーション"go"を遂行する。図34は、エージェント150Aによるオペレーション"go"の遂行の前のネットワーク1500の状態を示している。

【0530】エージェント150Aはオブジェクト140A、140Bを所有し、エンジン132A中のプレイス220Aを占有している。従ってエージェント150A及びプレイス220Aは、エンジン132Aによって同時に解釈されるプロセスである。換言すれば、プレイス220A、エージェント150Aは、オペレーション"live"を同時に遂行する。エンジン132Bは、プレイス220B及びエージェント150Bを解釈する。エージェント150Bはプレイス220Bを占有している。

【0531】エンジン132Aの通信下部構造132A-CIは、通信リンク102AZによって、エンジン132Zの通信下部構造132Z-CIに接続されている。エンジン132Zはプレイス220Zを解釈する。通信下部構造132Z-CIは、通信リンク102ZBによって、エンジン132Bの通信下部構造132B-CIにも接続されている。この実施例ではネットワーク1500中に3つのコンピュータシステムが示されているにすぎないが、ネットワーク1500のコンピュータシステムの数には任意の数とすることができる。従って図34に示したコンピュータシステムは本発明の原理を例示するものにすぎず、図示した特別のネットワークによって本発明を限定するものではない。

【0532】この実施例においてエージェント150Aによるオペレーション"go"の遂行は、エンジン132

Zを経てエンジン132Aからエンジン132Bにエージェント150Aを転送することを必要とする。従って、オペレーション”go”にはエンジン132Aすなわちソースエンジン、エンジン132Zすなわち移送エンジン及びエンジン132Bすなわち目的地エンジンの側の処置を必要としている。

【0533】ソースプレイスを処理するエンジンによって遂行されるオペレーション”go”

図35は、エージェント150Aによるオペレーション”go”の遂行の直前のエージェント150Aの実行状態の一部を含む内部状態の一部を示している。あるプロセスの実行状態については以下に詳細に説明する。スタック1304は、エージェント150Aの実行状態の一部である。スタック1304は上部”T”にチケット1306を備えている。チケット1306については以下に詳細に説明する。オペレーション”go”の遂行によって消費されたアーギュメントは、スタック1304から復元（ポップ）されるので、スタック1304はオペレーション”go”のコンテキストにおいて”現在のスタック”である。プレイス220Aはエージェント150Aの一つのプロパティであり、これはエージェント150Aがプレイス220Aを占めていることを示している。

【0534】図36は、エージェント150Aによるオペレーション”go”の遂行の直後においてのエージェント150Aの実行状態の一部を含む内部状態の一部を示している。図36については以下に詳述する。

【0535】図37に示すフローチャート（ロジックフローダイアグラム）1400は、エンジン132A（図34）によって実行されるオペレーション”go”を示している。エンジン132Aは、オペレーション”go”の遂行が開始された時にエンジン132A内において実行されているので、ソースエンジンである。図37に示すフローチャートのそれぞれのステップについては、プレイス220A（図34）からプレイス220Bへの、エージェント150Aによるトリップのコンテキストにおいて説明される。

【0536】エージェント150Aによるオペレーション”go”の遂行において、エンジン132A（図34）は、アクセステストステップ1402（図37）においてオペレーション”go”の遂行を要求しているエージェントがエージェント150Aであるかどうかを定める。オペレーション”go”がエージェント150A（図34）以外のオブジェクトによってリクエストされたものであれば、処理はアクセスステップ1402（図37）から終了ステップ1404に移行しこの終了ステップでは、クラスの例外”Process not Current（プロセスは現在のものではない）”が放出され、オペレーション”go”を失敗に終わらせる。しかしオペレーション”go”がエージェント150A（図34）によってリクエストされたものであれば、処理はアクセステストステップ1

402（図37）から”canGo”テストステップ1406に移行する。

【0537】”canGo”テストステップ1406においてエンジン132A（図34）はエージェント150Aがオペレーション”go”を遂行してもよいのか否かを定める。”canGo”テストステップ1406においてエンジン132A（図34）はエージェント150Aのアトリビュート”Permit”に対して質疑する。前述したアトリビュート”Permit”はエージェント150Aの複数の属性の一つである。エージェント150Aの色々な特性についてはクラス”Process”に関連してアペンディクスAに詳述されている。エージェント150Aのアトリビュート”Permit”の質疑によってエージェント150Aのプロパティ”Permit”が生成される。特定のパーミット（許可）1612（図38）はエージェント150Aの特性”Permit”であり従ってエージェント150Aのアトリビュート”Permit”を質疑することによって生成される。この実施例において特性”Permit”はエージェント150Aの複数のプロパティの中の一つである。

【0538】パーミット1612（図39）は、アペンディクスAに詳細に説明されている幾つかのプロパティの中に、プロパティ”charge”、”canGo”、及び”age”（其々整数1702、ブーリアン1704及び整数1706である）を含んでいる。ブーリアンは、只二つの可能な値”true”又は”false”の内の一つを有するオブジェクトである。”canGo”テストステップ1406（図37）においてエンジン132A（図34）は、パーミット1612（図39）のアトリビュート”canGo”に対して質疑する、それによりパーミット1612のプロパティ”canGo”であるブーリアン1704を生成させ、ブーリアン1704を”true”と比較する。

【0539】ブーリアン1704が”false”の値を持っていたら、処理は”canGo”テストステップ1406（図37）から終了ステップ1408に移行する。エージェント150Aは、オペレーション”go”の遂行によって移行することが許されず、オペレーションは失敗に終わり、終了ステップ1408においてクラス”Permit Violated”の1メンバである例外を投出する。例外及びその分類はアペンディクスAに一層詳細に説明されている。

【0540】またブーリアン1704（図39）が”true”の値を有していたら、処理は”canGo”テストステップ1406（図37）からエフェクチュエイトムーブステップ1410に移行する。エフェクチュエイトムーブステップ1410については図40を参照として以下に詳細に説明する。処理はエフェクチュエイトムーブステップ1410（図37）から終了ステップ1412に移行しそこでオペレーション”go”は、エンジン1

3 2 Aに関する限り成功の内に終了する。

【0 5 4 1】前述したようにエンジン1 3 2 A（図3 4）によって遂行されるエフェクチュエイト ムーブステップ1 4 1 0は、フローチャート1 4 1 0（図4 0）によって示されている。フローチャート1 4 1 0の第1のステップ、すなわちルートエージェントステップ1 4 1 4において、エンジン1 3 2 A（図3 4）は、エージェント1 5 0 Aをトリップの目的点すなわち”移送の目的点”まで転送する為に使用されるエンジンを選出する。ここに使用される移送目的点という用語は、トリップに使用する次のエンジンであり、トリップの目的点と混同するべきではない。チケット1 3 0 6（図3 5）は、トリップの目的点としての1つのプレイスを定義するので、”トリップ目的点”は1つのプレイスである。

チケット1 3 0 6（図3 5）によって規定されたトリップの行程において、エージェント1 5 0（図3 4）は1つのエンジンから次のエンジンへと移送される。従って、”移送目的点”は1つのエンジンである。一般にトリップには、エージェントがトリップ目的点を含むエンジンに到達するまでにエージェントの何回かの転送を必要とすることが有り得る。

【0 5 4 2】ステップ1 4 1 4（図4 0）において、ウェイオブジェクトすなわちクラス”ウェイ”の1つのメンバ（移送の目的点を規定する）が生成される。移送目的点は一例として（i）現にエージェント1 5 0 Aを含むエンジン、すなわちエンジン1 3 2 A（この場合エージェント1 5 0 Aは転送されない）、（i i）エンジン1 3 2 Aを含む領域内の別のエンジン、又は（i i i）別の領域のエンジンで有り得る。前記（i i）又は（i i i）のどちらの可能性においても、エージェント1 5 0 Aは後述するように移送目的点に向かって転送される。

【0 5 4 3】”領域”という用語は用語集に規定されている。エンジンを複数の領域に分類するのはエージェントのルーティング（移動）において顕著である。それは典型的には各々のエンジンが他の領域内のエンジンに比べて同一の領域内のエンジンについてより多くの詳細な情報を有しているからである。これは単一の領域のエンジンが単一人又は組織によって構成されてからであり、この人又は組織はその為領域の”供給者”と呼ばれる。従ってある領域の各々のエンジンには、その領域の他のエンジンについての詳細な情報が与えられることができる。それはこうした全てのエンジンが単一の供給者によって供給されているからである。

【0 5 4 4】エンジンを領域に分類することは、大型の複雑なネットワークにおいてエージェントをルーティングする上に有用である。あるエージェントが複数の領域の間を移動する場合、移送目的エンジンについての情報をソースエンジンが有していることは不可欠ではない。どの領域にエージェントが移動しているかをソース

エンジンが定めることができ、そしてそのエージェントをその領域内のあるエンジンに向かって移送させることができれば十分である。トリップの目的点であるプレイスを含む領域にエンジンが移送されたら、エージェントは一層容易にそのプレイスを含むエンジンに向かってルーティングさせることができる。

【0 5 4 5】フローチャート1 4 1 4（図4 3）は、ルートエージェントステップ1 4 1 4（図4 0）において実行される各ステップを示し、以下に一層詳細に説明される。

【0 5 4 6】処理はルートエージェントステップ1 4 1 4からアイソレート エージェントステップ1 4 1 6に移行し、そこでエージェント1 5 0 A（図3 4）が単離される。プロセスの単離についてはアペンディクスAに詳細に説明されており、その説明は引用によって本明細書の一部分となる。簡単に説明するとエージェント1 5 0 Aは、（i）エージェント1 5 0 A内の全ての参照を他のプロセス及びこれらの他のプロセスによって所有されているオブジェクトに対してポイドし、（i i）他の全てのプロセス内においてエージェント1 5 0 A及びエージェント1 5 0 Aによって所有されるオブジェクトに対する参照をポイドすることによって単離される。処理は単離エージェントステップ1 4 1 6（図4 0）からエクシットステップ1 4 1 8に移行する。

【0 5 4 7】エクシットステップ1 4 1 8において、プレイス2 2 0 A（図3 4）は、エンジン1 3 2 Aのリクエストによってオペレーション”exiting”を遂行することによって、エージェント1 5 0 Aが離去したことを知る。オペレーション”exiting”については以下に詳細に説明する。処理はエクシットステップ1 4 1 8からNEXT HOP IS HEREテストステップ1 4 2 0に移行する。

【0 5 4 8】NEXT HOP IS HEREテストステップ1 4 2 0においてエンジン1 3 2 A（図3 4）は、ルートエージェントステップ1 4 1 4（図4 0）において定められた移送目的点を現在のエンジンすなわちエンジン1 3 2 A（図3 4）と比較する。移送目的点が現在のエンジンすなわちエンジン1 3 2 Aであった場合、処理はNEXT HOP IS HERE テストステップ1 4 2 0（図4 0）からエージェント配送ステップ1 4 2 2に移行する。図3 4、図4 1、図5 0、図5 1、図5 3の例の場合、エージェント1 5 0 Aは、以下に説明するようにエンジン1 3 2 Zに移送される。従って移送目的点は現在のエンジンすなわちエンジン1 3 2 Aではなく、処理はエージェント配送ステップ1 4 2 2（図4 0）には移行しない。しかしエージェント配送ステップ1 4 2 2及び後続する各ステップについても以下に完全性を意図して説明する。

【0 5 4 9】エージェント配送ステップ1 4 2 2において、エージェント1 5 0 A（図3 4）はチケット1 3 0 6（図3 5）を満足するエンジン1 3 2 A内のプレイスに配送される。エージェント配送ステップ1 4 2 2（図

40)は、フローチャート1422(図49)によって示され、以下に詳細に説明される。処理はエージェント配送ステップ1422(図40)から第1の例外テストステップ1424に移行し、ここでエンジン132A

(図34)は、エージェント150Aの配送が例外を投出するか否かを定める。エージェント配送ステップ1422(図40)において例外が投出されない場合、処理は第1の例外テストステップ1424から終了ステップ1428に移行する。終了ステップ1428はムーブステップ1410を完全にするので、処理は、前述した終了ステップ1412(図37)に移行する。

【0550】エージェント配送ステップ1422(図40)において例外が投出された場合には、処理は第1の例外テストステップ1424からステップ1426に移行し、ここでエンジン150Aは、フローチャート1422(図49)に従ってパーガトリに配送される。パーガトリは各々のエンジン内の一つのプレイスでありプロセスへのイングレス(進入)を拒絶しない。パーガトリに入る際、エージェント150Aのローカル許可は著しく制限することができる。例えば、ローカル許可のプロパティ"canGo"は"true"にセットされ、"canCharge"、"canCreate"、"canDeny"、"canGrant"、"canRestart"、"canSend"、及び"canTerminate"の各プロパティは"false"にセットされる。プロパティ"charges"及び"age"は、エージェント150Aをパーガトリに送った例外をエージェントが検出し分析することができそしてオペレーション"go"の遂行によって別のプレイスに移動することができる程度に大きな値に設定する。許可に関するプロパティは、アペンディックスAに詳細に説明されている。

【0551】処理はステップ1426から終了ステップ1428に移行して、ムーブ移行し、さらにエフェクティブムーブステップ1410(図37)に移行し、前述したように成功のうちに終了する。

【0552】前述したように、ルートエージェントステップ1414(図40)において定めた移送目的点が現在のエンジンすなわちエンジン132Aであった場合には、処理はnext hop is here テストステップ1420(図40)からエージェント配送ステップ1422に移行する。逆に移送目的点が現在のエンジンでなかった場合、処理は、next hop is here ステップ1420から目的点形成ステップ1430に移行する。目的点形成ステップ1430において、移送目的点を特定する目的点、オブジェクトが形成される。処理は目的点形成ステップ1430からエージェント符号化ステップ1432に移行する。

【0553】エージェント符号化ステップ1432において、エンジン132A(図34)は、アペンディックスBに含まれるテレスク립ト符号化規則の規則にしたがってエージェント150Aを符号化する。アペンディ

ックスBは本明細書の一部分であり、引用によって全体として本明細書と一体化される。エージェント150Aの符号化によって、(i)エージェント150A、(i i)オブジェクト140A、140Bを含むエージェント150Aが所有するすべてのオブジェクト、及び(i i i)エージェント150A及びエージェント150Aの所有する全てのオブジェクトがそのメンバとなっている複数のクラスが標準化バイナリ形態において符号化されたエージェント150A-E(図41)として表される。目的点形成ステップ1430(図40)において形成された目的点オブジェクトすなわち目的点オブジェクト150A-E-D(図42)は符号化されたエージェント150A-Eに含まれ、以下に詳細に説明される。図41に示すように、符号化されたエージェント150A-Eは、エンジン132Aの通信下部構造132A-CIに記憶される。エージェント150A(図34)は、以下に詳細に説明されるように符号化されたエージェント150A-Eの移送が完了するまでエンジン132Aによって保持される。

【0554】処理はエージェント符号化ステップからトランスファアウトステップ1434に移行する。トランスファアウトステップ1434では、エンジン132A(図41)は、目的点形成ステップ1430(図40)において形成された目的点オブジェクトに従って、符号化されたエージェント150A-Eの移送を開始する。この例では目的点オブジェクトは、符号化されたエージェント150A-Eの移送目的点としてエンジン132Z(図41)を特定する。符号化されたエージェント150A-Eが、通信下部構造132A-CIから通信リンク100AZを経てエンジン132Zの通信下部構造132Z、CIに通常の2進データとして移送される。通信下部構造132A-CIと132Z-CIとの間のデータの転送は、アペンディックスC、Fに詳細に記述されている。これらのアペンディックスは本明細書の一部であり、全体として引用によって本明細書の一部とされている。前述したように符号化されたエージェント150A-Eの移送は、トランスファエージェントアウトステップ1434において開始される。このフローチャート1410(図40)による、エンジン132Aのプログラム部分132A-Pによる処理には通信下部構造132A-CIと132Z-CIとの間の符号化されたエージェント150A-Eの移送が続けられる限り継続される。

【0555】処理はトランスファアウトステップ1434から第2の例外テストステップ1436に移行し、ここでエンジン132Aは、トランスファアウトステップ1434が例外を投出したか否かをすなわち符号化されたエージェント150A-Eの移送の開始が失敗したかどうかを定める。トランスファアウトステップ1434が失敗した場合、処理は、エージェント150Aが前述し



たようにパーガトリに要求された第2の例外テストステップから移行する。逆にトランスファアウトステップ1434が成功した場合、処理は第2の例外テストステップ1436からステップ1438に移行する。

【0556】ステップ1438においてエンジン132A（図41）が、継続中のトランスファのリストにエージェント150Aを付加する。エージェント132Aは、エージェント150Aの移送トランスファの間継続中のトランスファのリストに保持されている。トランスファ（移送）が終了したら、エージェント150Aは、継続中のトランスファのリストから除かれ、エンジンによって廃棄される。エージェント150Aがなお継続中のトランスファのリストにある間にエージェント150Aの許可が終了した場合、すなわち、エージェント150Aの実際のエイジがエージェント150Aの有効許可のプロパティ“age”に到着した場合には、符号化エージェント150A-Eのトランスファはアボートされる。エージェント150Aは継続中のトランスファのリストから除かれ、エージェント150Aは、クラス“Permit Expired”の例外を投出することによって、オペレーション“go”を成功のうちに遂行することは失敗に終わる。プロセスの“有効許可”はアベンディクスに詳細に説明されている。処理はステップ1438から終了ステップ1428に移行し、そこでフローチャート1410（図40）、したがってエフェクチエートムーブステップ1410（図37）が終了する。

【0557】前述したようにエンジン132A（図34）は、ルートエージェントステップ1414において、エージェント150Aの移送目的点を定める。フローチャート1414（図43）は、ルートエージェントステップ1414（図40）において実行されるステップを示している。

【0558】フローチャート1414（図43、図44）の以下の説明は、（図34、図50）の例の範囲を越えた状況を考慮している。したがってフローチャート1414（図43、図44）は、図34、図41、図50、図51、図53の例とは無関係に説明される。しかし説明のための枠組みを与えるために、フローチャート1414（図43、図44）はエージェント150A（図34）のコンテキストによって説明される。このエージェント150Aは、エンジン132A内のプレイス220Aに発したトリップを行うためにオペレーション“go”を遂行する。フローチャート1414（図43、図44）の説明は、（i）エージェント150Aがエンジン132A（図34）以外のエンジンに移送されるステップ又は（ii）エージェント150Aがあるネットワーク（エンジン132A、132Z、132Bが同一の領域であるネットワーク）に移送されるトリップには限定されない。さらに、以下の説明は、エージェント150A（図34）のトリップを規定するチケット1

306（図35）を、ウェイ、手段、テレネーム又は供給者の指定を含むか、又は含まないように制限するものでもない。チケット1306（図35）は、フローチャート1414（図43、図44）のコンテキストにおいて、エンジン132A（図34）内のあるプレイス、

（ii）エンジン132Aと同一の領域内のエンジン内のあるプレイス、又は（iii）エンジン132Aを含む領域と異なったある領域内のあるエンジン内のあるプレイスへのトリップを規定することができる。

【0559】オペレーション“go”についてチケット1306（オペレーション“go”のアーギュメントとして消費されている）がトリップの目的点及びその他の特徴を特定化するものであることを想起されたい。フローチャート1414においてエンジン132A（図34）は、テストステップ1440において、チケット1306（図13、図45A）のアトリビュート“way”を質疑することによって、チケット1306のプロパティ“way”を生成させる。チケット1306の特性“way”は、ウェイ1820（図45A）である。チケット1306のプロパティ“way”はオプションであり、したがって0（図示せず）とすることもできる。

【0560】チケット1306のプロパティ“way”が0であった場合、処理はテストステップ1440から、ticket has provider テストステップ1460に移行する。このステップ1460については、以下に詳細に説明する。逆にチケット1306のプロパティ“way”が一つのウェイを持つ場合すなわち0でない場合、処理はテストステップ1440からステップ1442に移行しこのステップにおいてエンジン132A（図34）はウェイ1820（図45A）アトリビュート“name”を質疑することによって、ウェイ1820のプロパティ“name”であるテレネームを生成させる。またステップ1442（図43）においてエンジン132A（図34）は、そのテレネームのプロパティ“authority”を生成させ、コピーする。アベンディクスAに説明されるようにオクテットストリングによって表される、結果としたオーソリティは、後述するように、移送目的点を定めるために用いられる。

【0561】処理はステップ1442からway has means テストステップ1442（図43）に移行し、このステップ1444でウェイ1820（図46）のアトリビュート“means”が質疑されることによって、手段オブジェクト1822であるウェイ1820のプロパティ“means”が生成される。プロパティ“means”はオプションであるため、0（図示しないとしてもよい。）手段オブジェクトすなわちクラス“Means”のメンバは、

（i）コンピュータ間通信媒体及び（ii）この媒体を経て特定のエンジンに到達すべき移送インストラクションを特定することによってエンジンを特定するオブジェクトである。一例として手段オブジェクトは、公共の交

換される電話ネットワーク（PSTN）を特定し、エージェントを目的点エンジンに移送するための特定のモデムの電話番号を含むモデムインストラクションを特定することができる。手段オブジェクトは目的点エンジンにエージェント150Aをルーティングするのに必要なすべての情報を含むため、チケット1306の他のプロパティはチケット1306が手段オブジェクトを含む場合には無視される。

【0562】したがってウェイ1820のプロパティ"means"が0でない場合、すなわち手段オブジェクト1822（図46）である場合、処理はway has means

テストステップ1444（図43）からステップ1446に移行する。ステップ1446においてウェイ1822は移送目的点を規定するウェイであるとして指定される。なお手段オブジェクト1822は、最初のホップの目的点すなわちエージェント150Aの最初のトランスファの目的点（図34）であるエンジンを規定するものであり、必然的にエンジンではなく一つのプレイスでありチケット1306（図45A）によって規定されるトリップの目的点であるエンジンを規定するのではない。処理はステップ1446から終了ステップ1448に移行し、ここでフローチャート1414すなわちルートエージェントステップ1414（図40）の処理は成功の内に終了する。

【0563】逆にウェイ1820（図46）のプロパティ"means"が0である場合、ウェイ1820は手段オブジェクトを含まず、処理はway has means テストステップ1444（図43）からticket has name テストステップ1450に移行する。ticket has name テストステップ1452においてエンジン132A（図34）がテレネーム1818である、チケット1306（図45A）のプロパティ"destinationName"を発生させ、テレネーム1818を0と比較する。チケット1306のプロパティ"destinationName"はオプションであるので、チケット1306のプロパティ"destinationName"は0（図示しない）とすることができる。

【0564】チケット1306のプロパティ"destinationName"が0である場合、処理はticket has means テストステップ1450からticket has provider テストステップ1460に移行する。このテストステップ1460については以下に説明する。その逆に、チケット1306のプロパティ"destinationName"がテレネーム1818である場合には処理はticket has name テストステップ1450からステップ1452に移行する。ステップ1452においてエンジン132A（図34）はトランスファ目的点へのエージェント150Aの移送通路を規定するウェイオブジェクトを発生させる。エンジン132Aは、以下に詳細に説明するファインダと相談し、ステップ1442（図43）において発生されコピーされたオーソリティーに関連された領域へのウェイ

オブジェクトと、その名前がテレネーム1818（図45A）に等しいオーソリティー内のプレイスを発生させる。ファインダの使用によってこのようなウェイオブジェクトを発生させることについては以下に詳細に説明する。

【0565】処理はステップ1452から"way out is here" テストステップ1454に移行し、このテストステップ1454においてエンジン132A（図34）は、ステップ1452（図43）において発生させたウェイオブジェクトがエンジン132A（図34）へのトランスファを規定するか否かを定める。生成させたウェイオブジェクトがエンジン132Aへのトランスファを規定していない場合、処理は"way out is here" テストステップ1454から終了ステップ1456に移行し、この終了ステップにおいてフローチャート1414のすなわちルートエージェントステップ1414（図40）の処理は成功の内に終了する。

【0566】その逆に生成させたウェイオブジェクトがエンジン132A（図34）へのトランスファを規定している場合、処理は、"way out is here" テストステップ1454から1458に移行し、このステップ1458においてエンジン132Aは、エージェント150Aがエンジン132Aに移送されるべきことを示すフラグを立てる。処理はステップ1458から、以下に詳細に説明する。第2のticket has name テストステップ1480に移行する。

【0567】前述したようにチケット1306のプロパティ"way"が0である場合には、処理は、ticket has name テストステップ1440からticket has provider テストステップ1460に移行する。ticket has provider テストステップ1460においてエンジン132Aは、テレアドレス1814（図45A）である。チケット1306のプロパティ"destinationAddress"を発生させ、テレアドレス1814のプロパティ"provider"を発生させる。チケット1306のプロパティ"destinationAddress"又はテレアドレス1814のプロパティ"provider"のどちらかが0であった場合、処理はticket has provider テストステップ1460から、以下に詳細に説明する第2のticket has name テストステップ1480に移行する。その逆に、チケット1306（図45A）のプロパティ"destinationAddress"及びテレアドレス1814のプロパティ"provider"が0でない場合には、チケット1306は、テレップの目的点を含む領域を指定し、処理はticket has provider テストステップ1460からステップ1462に移行する。

【0568】ステップ1462においてエンジン132A（図34）は、以下に詳細に説明するように、ファインダと相談し、チケット1306のプロパティ"destinationAddress"のプロパティ"provider"によって示された領域のエンジンへのトランスファを規定するウェイ

オブジェクトを発生させる。処理はステップ1462

(図43)からfirst found テストステップ1464に移行し、このステップ1464においてエンジン132A(図34)は、ステップ1462においてウェイオブジェクトが成功の内に生成されたか否かを定める。ステップ1462においてウェイオブジェクトが生成された場合、処理はfirst found テストステップ1464から第2の"way out is here" ステップ1472(以下に詳細に説明する)に移行する。逆にステップ1462において、ウェイオブジェクトが生成されなかった場合には、処理は、first found テストステップからステップ1466に移行する。

【0569】ステップ1466において、エンジン132A(図34)は、以下により詳細に説明するように、ファインダと相談し、チケット1306のプロパティ"destinationAddress"であるテレアドレスのプロパティ"routingAdvice"の一つのアイテムによって示されるある領域へのトランスファを規定するウェイオブジェクトを発生させる。テレアドレスの特にプロパティ"provider"及び"routingAdvice"の各プロパティはアベンディクスAに一層詳細に説明されており、この説明は引用によって本明細書の一部とされる。簡単に説明すると、テレアドレスのプロパティ"routingAdvice"のアイテムは、チケット1306(図45A)によって規定されるトリップの目的点にエージェント150Aを移送させることができるとテレアドレスの作成者によって信じられている1以上の領域の供給者(プロバイダ)を指示する。処理はステップ1466(図43)から、第2のfound テストステップ1468に移行し、このテストステップ1468において、エンジン132A(図34)は、ウェイオブジェクトがステップ1466において成功の内に生成されたか否かを定める。ステップ1466においてウェイオブジェクトが生成されなかったら、処理は第2のfound テストステップ1468から終了ステップ1470に移行し、この終了ステップにおいてクラス"目的点使用不可能"の例外が投出される。その逆にステップ1466においてウェイオブジェクトが生成された場合には、処理は第2の found テストステップ1468から第2の"way out is here" テストステップ1472に移行する。

【0570】第2の"way out is here" テストステップ1472において、エンジン132A(図34)は、ステップ1462又はステップ1466(図43)において発生させたウェイオブジェクトが、エンジン132A(図34)へのトランスファを規定しているか否かを定める。生成されたウェイオブジェクトがエンジン132Aへのトランスファを規定していない場合には、処理は、第2の"way out is here" テストステップ1472から終了ステップ1470に移行し、この終了ステップにおいて論理フローダイアグラム1414すなわち

ルートエージェントステップ1414(図40)の処理は成功の内に終了する。

【0571】その逆に生成したウェイオブジェクトがエンジン132A(図34)へのトランスファを規定している場合には、処理は、第2の"way out is here" テストステップ1472からステップ1476に移行し、このステップ1476においてエンジン132A(図34)は、チケット1306によって規定されるトリップの目的点を特定化するテレネームを、チケット1306のプロパティ"destinationAddress"であるテレアドレスのプロパティ"location"から導くことになる。

【0572】エンジン132Aは、便利で効率的でない方法によってテレネームを導いてもよい。一例としてエンジン132Aは、キーがオクテットストリング例えばオクテットストリング2000K1であり値がテレネーム例えばテレネーム2000V1であるディクショナリ2000(図47)のようなテーブルを使用しても良い。一例として、テレアドレス1814(図45A)のプロパティ"location"がオクテットストリング2000K1(図47)に等しい場合には、テレネーム2000V1は、チケット1306(図45A)によって規定されたトリップの目的点であるプレイスを特定化するテレネームとしてエンジン132Aによって生成される。

【0573】アドレスは、ある領域内の番地に、その領域のエンジンのデザイン及びインプリメントによってデザインされインプリメントされたアドレッシングスキームに従って割り当てられている。開示されたインストラクションセットは、特定のアドレッシングスキームを指定しない。各々の領域は、その特別の領域のもっとも効率的で便利なアドレス指定スキームを自由にインプリメントすることができる。従って世界の各々の主権国家がその自身のアドレス指定スキームをその郵便サービスを提供する際にデザインレイアウトするのと同様に、各々の領域は、その領域のための独特のアドレス指定スキームを自由にインプリメントすることができる。勿論2以上の領域内において使用されるアドレス指定スキームは単一でなくても良い。

【0574】アドレス指定スキームをインプリメントする際に、ある領域のエンジン(複数)は、1以上のプレイスを含むロケーションにテレアドレスを割り当てる。テレアドレスを割り当てる際にエンジンはそのテレアドレスによって特定されたロケーション内の1以上のプレイスを特定するテレネームを発生させそして記録する。

換言すれば、テレアドレスを割り当てるエンジンは、それぞれのテレアドレスが割り当てられたプレイスのテレネームにテレアドレスを連係させる情報を保持している。このようにしてエンジン例えばエンジン132A(図34)は、特定化されたテレアドレスを有するプレイスのテレネームを発生させることができる。このよ

うなテレネームを記録しそして発生させる特定のメカニズムは、ある与えられた領域のエンジンをデザインインプリメントする個人又は組織に一任される。

【0575】処理はステップ1476からステップ1478に移行し、このステップ1478においてエンジン132A(図8)は、エンジン132Aを含む領域内のあるエンジンにエージェント150Aが移送されるべき事を示すフラグをたてる。処理はステップ1478(図43)から第2のticket has name テストステップ1480(図43)に移行する。さらに処理は、ステップ1458(図43)及びticket has provider テストステップ1460(これらのステップはどちらもすでに詳細に説明されたものである)から第2のticket has name テストステップ1480に移行する。

【0576】第2のticket has name テストステップ1480(図44)において、エンジン132A(図34)は、チケット1306(図45A)のプロパティ"destinationName"が0か、又はテレネーム1818であるかを定める。チケット1306のプロパティ"destinationName"がテレネーム1818である場合には処理は、第2のticket has name テストステップ1480(図44)からステップ1482に移行し、このステップ1482においてエンジン132A(図34)はテレネーム1818(図45A)のコピーを生成させ、そのコピーは、移送目的点のテレネームとして、ステップ1476(図43)において導出されたテレネームに変わる。処理はステップ1482(図44)からステップ1483に移行する。また、チケット1306のプロパティ"destinationName"が0である場合には、処理は第2のticket has name テストステップ1480から1483に直接に移行する。

【0577】エンジン132A(図34)は、ステップ1483において、以下に詳細に説明するように、ファインダと相談し、前述したようにステップ1476(図43)又はステップ1482(図44)のどちらかで生成させたテレネームによって指示されるプレイスへのトランスファを規定するウェイオブジェクトを発生させる。処理はステップ1483からテストステップ1484に移行し、このテストステップにおいてエンジン132A(図34)は、ステップ1458(図43)がトランスファの目的点エンジン132A(図34)であることを示すフラグを立てる。フラグが立てられていない場合、処理はテストステップ1484から、後述するテストステップ1490に移行する。その逆に、フラグが立てられていた場合、エージェント150A(図115)は、エンジン132A内のあるプレイスにトランスファさせる必要があり、処理はテストステップ1480(図44)から第3の"way out is here" テストステップ1486に移行する。

【0578】第3の"way out is here" テストステッ

プ1486においてエンジン132A(図34)は、ステップ1483(図44)によって発生させたウェイオブジェクトがエンジン132A(図34)へのトランスファを規定しているか否かを定める。生成させたウェイオブジェクトはエンジン132Aのトランスファを規定していない場合には、処理は第3の"way out is here" テストステップ1486(図44)から終了ステップ1488に移行し、この終了ステップにおいて、クラス"目的点使用不可能"の例外が投出され、論理フローダイアグラム1414(図43、図44、)すなわちルートエージェントステップ1414(図40)の処理が終了する。その逆に、生成されたウェイオブジェクトがエンジン132A(図34)内のあるプレイスへのトランスファを規定しているならば、処理はテストステップ1486(図44)からテストステップ1492に移行する。

【0579】テストステップ1490においてエンジン132A(図34)は、トランスファの目的点エンジン132A(図34)を含む領域であることを指示するフラグを立てる。このフラグが立てられていない場合、処理はテストステップ1490から終了ステップ1496に移行し、この終了ステップにおいて、論理フローダイアグラム1414(図43、図44)すなわち、ルートエージェントステップ1414(図40)による処理が成功の内に終了する。その逆にフラグが立てられていない場合、エージェント150A(図34)は、エンジン132Aを含む領域内のあるプレイスに移行させる必要があり、処理はテストステップ1490(図44)からway out is this region テストステップ1492に移行する。エンジン132A(図34)は、way out is this region テストステップ1492において、ステップ1483において発生させたウェイオブジェクト

(図44)がエンジン132A(図34)を含む領域内のあるプレイスへのトランスファを規定しているか否かを定める。発生させたウェイオブジェクトがエンジン132Aを含む領域内のあるプレイスへのトランスファを規定していない場合には、処理は、way out is this region テストステップ1492(図44)から終了ステップ1494に移行する。この終了ステップにおいてはクラス"目的点使用不可能"の例外が投出され、論理フローダイアグラム1414(図43、図44)すなわちルートエージェントステップ1414(図40)による処理が終了する。

【0580】エンジン132A(図132)は、ステップ1452、1462、1466及び1483(図43)について以上に説明したように、チケット1306(図35)によって規定されたトリップの目的点へのトランスファにエージェント150Aを差し向けるために、ファインダを使用する。ファインダはエージェントのトランスファの目的点を定めるために使用される。一

例としてチケット1306（図35）は、トリップの目的点として、エンジン132B内のプレイス220B

（図34）を規定する。しかしエージェント150Aは、エンジン132Bに到達するためには、最初にエンジン132Zに移送されなければならないので、エンジン132Zはトランスファの目的点となる。ファインダ2050（図48）は、第2のテレネームによって特定化されるプレイスに向かうトランスファの目的点であるエンジンのテレネームを発生させるために用いられる。指示指令によれば、生成されたテレネームは、エンジンがエンジンによって処理されるエンジンプレイスを特定することによってエンジンを特定する。第2のテレネームは、特定のプレイス又はオーソリティを特定化しそれによって特定化されたオーソリティの全てのプレイスを特定化する。テレネームはアペンディックスAに一層詳細に説明されている。

【0581】ステップ1452（図43）のコンテキストにおいて、第2のテレネームは、テレネーム1818（図45A）であり、このテレネームは、チケット1306のプロパティ“destinationName”である。ファインダ2050（図48）がテレネーム1818に関する情報を含まない場合、ステップ1452（図43）コンテキストにおいて第2のテレネームは、前述したようにステップ1442において発生されコピーされるオーソリティを特定化するテレネームである。第2のテレネームは、ステップ1462のコンテキストにおいてチケット1306のテレアドレス1814（図45A）のプロパティ“provider”において特定されたオーソリティを特定化する。第2のテレネームは、ステップ1466

（図43）のコンテキストにおいて、チケット1306のテレネーム1814（図45A）のプロパティ“routingAdvice”であるリストの一つのアイテムにおいて特定化されたオーソリティを特定化する。ステップ1483（図44）のコンテキストにおいて、第2のテレネームは、チケット1306のプロパティ“destinationName”が0でない場合、チケット1306のテレネーム1818（図45A）であるか、又はその他の場合において、ステップ1476（図43）において生成したテレネームである。ファインダ2050（図48）は、第2のテレネームがチケット1306のテレネーム1818（図45A）であるステップ1483（図14）のコンテキストにおいて以下に説明される。

【0582】ステップ1452（図43）、1462、1466及び1483（図44）の各々において、ファインダ2050（図48）の使用によって生成させたテレネームは、生成させたテレネームによって特定化されるエンジンへのトランスファを規定するウェイオブジェクトを形成するために用いられる。

【0583】エンジン132Aは、ファインダ2050（図48）の使用によって、エージェント150Aを転

送すべきエンジン、すなわち”移送目的点”を定める。ファインダ2050は、トリップの目的点からトリップの目的点に向かってエージェント150Aを移動させるトランスファ目的点をエンジン132Aによって定めることを可能とするものであれば、どんな構造であってもよい。一実施例によれば、ファインダ2050は、キーがテレネームであり、値がやはりテレネームであるディクショナリである。キー例えばテレネーム2050K1-2050K6は、オーソリティを特定化する。値、例えばテレネーム2050V1-2050V6は、各々の対応するオーソリティによって特定されたプレイスに到達する際に経過するエンジン（複数）を特定する。ファインダ2050のテレネーム2050K1（図48）がテレネーム1818のものと同一のオーソリティを持つ場合には、テレネーム2050V1は、トランスファ目的点として、エンジン132Zを特定化する。

【0584】本発明の一実施例によれば、ファインダ2050のキーは、テレネーム2050V7に組み合わせられたニル（零）2050K7を含む。

【0585】テレネーム2050V7は、ネットワーク1500（図34）及びネットワーク1500が直接又は間接に接続され得る複数のネットワークの実質的な部分に関する情報を含むエンジンを特定化する。したがって、テレネーム2050V7によって特定化されたエンジンは、チケット1306（図45A）を満足するトリップ目的点に向かってエージェント150Aをルーティングする際に最も成功する確率の高いエンジンである。ファインダ2050（図48）のキーであるテレネームがテレネーム1818と同一のオーソリティのものでない場合、ニル2050K7に組み合わせられたテレネーム2050V7（図48）は、トランスファの目的点を特定化する。次の例は例示的である。

【0586】小型のパーソナルコンピュータシステム内においてエンジン132Aが実行され、従って、エンジン132Aがただ一つのオーソリティのプレイス（複数）を解釈するものと想定する。さらに、エンジン132Z（図34）が大型のマルチユーザメインフレームコンピュータシステム内において稼働し、このコンピュータシステムが多くの別々のオーソリティのプレイス（複数）を解釈し、このコンピュータシステムに多くの異なったオーソリティの多くのエージェントが移動するものと想定する。このような場合、エンジン132Aのファインダは、非常に小型になり、特別のオーソリティのプレイスへのエージェントの転送についてほとんど情報をもたらさないであろう。しかし、エンジン132Zのファインダは、より広範で包括的であると思われるので、特別のオーソリティのプレイスに向かって特別なエージェントをルーティングすることについて情報を提供する可能性がより多く存在する。このような場合、エンジン132Aのファインダは、トリップの目的点プレイスの

オーソリティに関する情報をエンジン132Aのファインダが含まない場合にエンジン132Zをトランスファの目的点として特定化するテレネームをニルと組み合わせる。

【0587】ファインダ2050（図48）との相談によってトランスファの目的点を定めることが成功しなかった場合、エンジンは、エンジンの供給者によって選定されたインクリメンテーションに従って、次のポリシー又はその他のポリシーのどれかをインクリメントすることができる。

【0588】チケット1306（図45A）がサイテーション1816を含む場合、エンジン132A（図34）は、エージェント150Aの現在のプレイスにあるテレネームを発生させ、以下に説明するステップに従って、引用されたクラスのプレイスを見出すように努めるか、又は（ii）例外を投出してテレネームを発生させないことによってチケット1306（図45A）をあいまいであるとして拒絶し又は（iii）あいまいに特定化されたトリップの目的点として指定されたプレイスのテレネームを生成させることができる。同様に、チケット1306がサイテーションを持たない場合、エンジン132A（図34）は、（i）例外を投出しテレネームを発生させないことによって、チケット1306（図45A）をあいまいとして拒絶し又は（ii）あいまいに特定化されたトリップの目的点として指定されたプレイスのテレネームを発生させることができる。

【0589】図34、図41、図50、図51、図53の例では、エンジン132A（図34）は、ファインダ2050（図47）に相談し、エンジン132Bのプレイス220B（図34）に到達するためにはエージェント150Aはエンジン132Zに移送させるべきことを定める。従って、エンジン132Zのエンジンプレイスは、エージェント150Aの移送目的点であり、エンジン132Zへのエンジン150Aの移送を規定するウェイオブジェクトが生成される。

【0590】このようにフローチャート1414（図43、図44）に従った処理によって、チケット1306によって規定されたトリップの目的点に向かってエージェント150Aを移動させるためのエージェント150A（図34）のトランスファを規定するウェイオブジェクトが生成される。

【0591】フローチャート1410（図40）のコンテキストにおいて前述したように、エージェント150A（図34）は、place in deliver agentステップ1422（図40）中のあるプレイスに配送される。図34、図41、図50、図51、図53の実施例においてエージェント150Aは、エンジン132A内のいかなるプレイスにも配送されないことを想起されたい。しかし、記述を完全にするために、エンジン132A内のプレイスへのエージェント150Aの配送について説明す

る。エージェント150A（図34）は、エンジン132A内のプレイスに、このプレイスがチケット1306（図35）によってトリップの目的点として規定される場合に配送される。あるプレイスへのエージェントの配送は、（i）そのエージェントを配送するべき特別なプレイスを選定すること、（ii）そのエージェントをそのプレイスの占有者として、及び（iii）オペレーション”go”をそのエージェントについて継続させること、を含む。エージェント配送ステップ1422は、フローチャート1422（図49）によって示されている。

【0592】フローチャート1422（図49）による処理は、for all current placesステップ1422Aにおいて開始される。for all current placesステップ1422A及び次のステップ1422Cは、現在のエンジン、すなわちエンジン132A（図34）によって現に処理されているプレイスの各々がステップ1422B、1422D及び1422E（図49）に従って処理される処理ループを規定する。フローチャート1422の以下の説明のコンテキストにおいて”サブジェクトプレイス”は、for all current placesステップ1422A及び次のステップ1422Cによって規定されるループのある特別の反復において処理されるプレイスである。このループの各々の繰り返し（反復）すなわちエンジン132A（図34）の各々の現在のプレイスについて、処理は、for all current placesステップ1422Aから、place satisfies ticketテストステップ1422Bに移行する。

【0593】現在のエンジンは、place satisfies ticketテストステップ1422Bにおいて、サブジェクトプレイスがチケット1306を満足するか否かを定める。

【0594】サブジェクトプレイスがチケット1306を満足しない場合、（i）処理は、place satisfies ticket テストステップ1422B（図49）から次のステップ1422Cを経てfor all current placesステップ1422A（ここで、ループの別の反復が開始される）に移行するか、または、（ii）処理は、以下に示す説明する終了ステップ1422Iに移行する。逆に、サブジェクトプレイスがチケット1306を満足する場合、処理は、place satisfies ticket テストステップ1422Bからエンタリングステップ1422Dに移行する。

【0595】以下に詳細に説明するように、あるプレイスは、オペレーション”entering”を成功の内に遂行することによってあるプロセス例えばあるエージェントへのインGRESを許容し、オペレーション”entering”の実行の間に例外を投出することによってプロセスへのインGRESを拒絶する。オペレーション”entering”の遂行については以下に一層詳細に説明する。

【0596】処理は、エンタリングステップ1422D

(図49)から例外テストステップ1422Eに移行し、テストステップ1422Eにおいて、現在のエンジンは、サブジェクトプレイスによるオペレーション”entering”の遂行が例外を投出したか否かを定める。サブジェクトプレイスによるオペレーション”entering”の遂行が例外を投出し、それによりサブジェクトプレイスへのエージェント150Aのイングレスを拒絶した場合、処理は、エンタリングステップ1422Eから次のステップ1422Cを経てfor all current placesステップ1422Aに移行し、for all current placesステップ1422A及び次のステップ1422Cによって規定されたループの別の反復が開始されるようにする。または、処理は、以下に説明する終了ステップに移行する。

【0597】その反対にサブジェクトプレイスによってオペレーション”entering”が成功の内に遂行され、サブジェクトプレイスへのイングレスをエージェント150A(図34)に対して許容した場合、処理は例外テストステップ1422E(図49)からステップ1422Fに移行する。エージェント150A(図15)は、ステップ1422Fにおいて、サブジェクトプレイスの占有者にされる。さらに、エージェント150Aがサブジェクトプレイスの占有者であることを示すために、エージェント150Aのある制御がセットされる。

【0598】処理は、ステップ1422Fから1422Gに移行し、このステップ1422Gにおいてエージェント150Aによって遂行されるオペレーション”go”が成功する。エージェント150Aは、前述したように、図34、図41、図50、図51、図53の例では、エンジン132Aのプレイスには配送されない。従って、エージェント150Aのためのオペレーション”go”は、以下に説明するようにエンジン132B(図50)のプレイス220Bにエージェント150A(図34)が配送されるまでは成功しない。処理は、ステップ1422G(図49)から終了ステップ1422Hに移行し、この終了ステップにおいて、フローチャート1422に従って、エージェント配送ステップ1422(図40)に従った処理が成功の内に終了する。

【0599】for all current places ステップ1422A(図49)及び次のステップ1422Cによって規定されるループに従って、エンジン132A(図15)中に存在するすべてのプレイスが処理され、終了ステップ1422Hには到達していない場合、すなわちエンジン132A(図34)中に存在する如何なるプレイスもチケット1306を満足させず、またオペレーション”entering”の成功した遂行を介してエージェント150Aの占有を許容しない場合、処理は、for all current placesステップ1422A(図49)から終了ステップ1422Iに移行する。終了ステップ1422Iにおいては、クラス”目的点使用不可能”の例外が投出され、

フローチャート1422、従ってエージェント配送ステップ1422(図40)は終了する。このように、エージェント配送ステップ1422は、トリップの目的点としてチケット1306(図35)によって特定されたプレイスにエージェント150A(図34)がエージェント配送ステップ1422によって配送されるかまたはクラス”目的点使用不可能”の例外が投出される。

【0600】前述したように、図34、図41、図50、図51、図53の実施例においては、エンジン132Zは、トランスファ目的点である。従って、エージェント150Aは、ステップ1432(図40)において符号化されたエージェント150A-2(図41)を形成するように符号化される。符号化されたエージェントは、トランスファアウトステップ1434(図40)において、エンジン132Z(図51)に移送される。

【0601】中間エンジンの視点から見たオペレーション”go”

前述したように”通信下部構造132A-CIから通信リンク102AZを経て通信下部構造132Z-CI

(図51)に至る符号化エージェント150A-2(図41)のトランスファは、トランスファアウトステップ1434(図40)において開始される。それぞれテレネーム2102、テレアドレス2104及びオクテットストリング2106であるプロパティ”name”、”address”、及び”data”を含む目的点150A-E-D

は、符号化されたエージェント150A-E(図42)に付加されている。テレネーム2102及びテレアドレス2104は、符号化エージェント150A-Eのトランスファ目的点を規定する。目的点150A-E-D(図42)によって規定されたトランスファ目的点は、132Z(図41)でもエンジン132Bでもよい。

【0602】本発明の一実施例によれば、目的点形成ステップ1430(図40)において形成された目的点150A-E-D(図42)は、符号化エージェント150A-Eのトランスファ目的点として、エンジン132Z(図41)を規定する。この場合において、通信下部構造132A-CIは、符号化エージェント150A-Eを132Zの通信下部構造132Z-CIに直接転送する。本発明の他の実施例によれば、目的点150A-E-D(図42)は、符号化エージェント150A-Eのトランスファ目的点としてエンジン132B(図41)を規定する。後者の場合に、通信下部構造132A-CIは、目的点150A-E-D(図42)に従って、エンジン132Bに対して設定された符号化エージェント150A-Eを通信下部構造132Z-CI(図41)又は132Z-CIを介してトランスファさせるための情報及びロジックを収納している。

【0603】前述したどちらの実施例においても、エンジン132Z(図51)が符号化エージェント150A-Eを受けると、エンジン132Zは、フローチャート



1400-I (図52) に示されたルートエージェントステップ1414 "トランスファイン" において定められたシステムオペレーションを遂行する。エンジン132Z (図51) は、ステップ1402-I (図52) において目的点150A-E-D (図42) へのトランスファを規定するウェイオブジェクトを発生させることによって、システムオペレーション "transferIn" の遂行を開始する。エンジン132Zは、ファインダ2050 (図48) について前述したように、エンジン132Z (図51) 内のファインダと相談することによって、ステップ1402-I (図52) においてウェイオブジェクトを発生させる。

【0604】処理はステップ1402-I (図52) から第4の "way out is here" テストステップ1404-Iに移行し、ここでエンジン132Z (図51) は、ステップ1402-I (図52) において発生させたウェイオブジェクトがエンジン132Z (図51) へのトランスファを規定しているか否かを定める。発生させたウェイオブジェクトがエンジン132Z (図51) をトランスファ目的点として特定していない場合、すなわち目的点150A-E-D (図42) がエンジン132B (図41) をトランスファ目的点として規定している場合、処理は第4の "way out is here" テストステップ1404-I (図52) から第2のトランスファアウトステップ1406-Iに移行する。第2のトランスファアウトステップ1406-Iにおいてエンジン132Z (図51) は、トランスファアウトステップ1434 (図40) について以上に説明した仕方と直接類似した仕方、目的点150A-E-D (図42) に従って符号化エージェント150A-Eの転送を開始する。

【0605】処理は第2のトランスファアウトステップ1406-Iからステップ1408-Iに移行する。ステップ1408-Iにおいて、エンジン132Z (図51) は、ステップ1438 (図40) について前述したように継続中のトランスファのリストに符号化エージェント150A-Eを付加する。処理はステップ1408-I (図52) から終了ステップ1410-Iに移行し、ここでシステムオペレーション "transferIn" は成功のうちに終了する。

【0606】ステップ1402-Iにおいて生成したウェイオブジェクトが、エンジン132B (図51) へのトランスファを規定している場合、符号化データ150A-Eは、生成されたウェイオブジェクトに従って転送される。生成されたウェイオブジェクトがエンジン132Zへの転送を規定している場合、処理は第4の "way out is here" テストステップ1404-I (図52) からステップ1412-Iに移行する。ステップ1412-Iにおいてエンジン132Z (図51) は、符号化されたエージェント150A-Eからチケット13

06 (図35、図45A) を抽出し、チケット1306のコピーを形成する。処理はステップ1412-Iからステップ1414-Iに移行し、ここでチケットコピーのプロパティ "way" がクリアーされる (帰零される)。処理はステップ1414-Iから第2のルートエージェントステップ1416-Iに移行し、ここでエンジン132Zは、符号化エージェント150A-Eのトランスファを規定するウェイオブジェクトを発生させる。ステップ1416-Iにおいて遂行されたプロセスは、フローチャート1414 (図43、図44) によって表わされたプロセスと同一である。第2のルートエージェントステップ1416-Iのコンテキストにおいて、フローチャート1414の上述した説明において使用されたチケット1306は、ステップ1412-Iにおいて作成したチケットコピーによって代えられ、フローチャートは、そのチケットコピーを用いて前述したように処理される。ステップ1416-Iが終了すると、処理は第5の "way out is here" テストステップ1418-Iに移行する。

【0607】第5の "way out is here" テストステップ1418-Iにおいて、エンジン132Z (図51) は、ステップ1416-I (図52) において発生させたウェイオブジェクトがエンジン132Z (図51) へのトランスファを規定しているか否かを定める。発生させたウェイオブジェクトが、エンジン132Zへのトランスファを規定している場合、処理は第5の "way out is here" テストステップ1418-I (図52) から、以下に詳細に説明するエージェント解号ステップ1428-Iに移行する。その反対に、生成させたウェイオブジェクトがエンジン132Z (図51) へのトランスファを規定していない場合、処理は第5の "way out is here" テストステップ1418-I (図52) から第2の目的点形成ステップ1420-Iに移行する。図34、図41、図50、図51、図53の実施例において、符号化エージェント150A-Eから抽出されコピーされたチケット1306 (図35) は、エンジン132Bのプレイス220Bへのトリップを規定する (図51)。従って処理は第2の目的点形成ステップ1420-Iに移行する。

【0608】目的点形成ステップ1420-Iにおいて、エンジン132Z (図51) は、目的点形成ステップ1430 (図40) について前述した仕方と直接類似した仕方、目的点150A-E-D (図42) を形成し、すなわち代替する。処理は第2の目的点形成ステップ1420-I (図52) から第3のトランスファアウトステップ1422-Iに移行する。第3のトランスファアウトステップ1422-Iにおいて、符号化されたエージェント150A-E (図51) は、第2のトランスファアウトステップ1406-I (図52) について前述したように、目的点150A-E-D (図4

2)に従って転送される。

【0609】処理は第3のトランスファーアウトステップ1422-Iからステップ1424-Iに移行する。ステップ1424-Iにおいてエンジン132Z(図51)は、ステップ1438(図40)について前述したように、継続中のトランスファーのリストに符号化されたエージェント150A-Eを付加する。処理は1424-I(図52)から終了ステップ1426-Iに移行し、ここでシステムオペレーション"transferIn"は成功のうちに終了する。図34、図41、図50、図51、図53の実施例において、符号化エージェント150A-Eは、前述したようにエンジン132Aから132Zへの符号化エージェント150A-Eのトランスファーと直接類似した仕方で、エンジン132Zからエンジン132B(図53)に移行する。

【0610】目的別エンジンの視野から見たオペレーション"go"

エンジン132Bが通信下部構造132A-CI(図53)中の符号化されたエージェント150A-Eを受けた時、エンジン132Bは、エンジン132Zについて前述した仕方と同様の仕方で、システムオペレーション"transferIn"を遂行する。エンジン132Bは、ステップ1402-I(図52)において目的点150A-E-D(図42)に従ってトランスファーを規定するウェイオブジェクトを発生させ、発生させたウェイオブジェクトが、第4の"way out is here"テストステップ1404-I(図52)中のエンジン132B(図53)へのトランスファーを規定しているか否かを定める。エンジン132B(図53)は、チケット1306(図35)によって規定されたトリップの目的点であるプレイス220Bを処理するので、生成されたウェイオブジェクトは、エンジン132B(図53)へのトランスファーを規定している。従ってエンジン132Bによる処理は、第4の"way out is here"テストステップ1404-I(図52)からステップ1412-Iに移行する。

【0611】ステップ1412-I、1414-I及び1416-Iにおいて、エンジン132Bは、符号化エージェント150A-Eからチケット1306(図35)を抽出してコピーし、チケットコピーのプロパティ"way"をクリアし、チケットコピーが今や以上の説明においてチケット1306の代りに使用されていることを除いてフローチャート1414(図43、図44)に従って符号化エージェント150A-Eを転送(ルーティング)する。第5の"way out is here"テストステップ1418-I(図52)において、エンジン132B(図53)は、第2のルートエージェントステップ1416-I(図52)において生成させたウェイオブジェクトがエンジン132B(図53)へのトランスファーを規定しているか否かを定める。エンジン1

32B(図53)はプレイス220Bを処理するので、生成されたウェイオブジェクトは、エンジン132B

(図53)へのトランスファーを規定する。従ってエンジン132Bによる処理は、第5の"way out is here"テストステップ1418-I(図52)からエージェント解号ステップ1428-Iに移行する。

【0612】エージェント解号ステップ1428-Iにおいて、エンジン132Bは、アペンディックスBに説明した符号化規則の逆の適用によって、通信下部構造132B-CI中の符号化エージェント150A-E(図53)からエージェント150Aを解号し、データ部分132B-D(図50)中にエージェント150Aを格納する。ネットワーク1500は異質でありうるため、エンジン132A(図34)内のエージェント150Aの形式は、エンジン132B(図50)内のエージェント150Aを表わすには不具合であることがありうる。エンジン132Aと132Bとの間にエージェント150Aを移送させるために標準化形式の符号化エージェント150A-Eを使用している限りにおいて、エンジン132Bは、エージェント150Aを移送させていない場合、エンジン132Bにとって最も好都合な任意の形態においてエージェント150Aを表現することができる。

【0613】エンジン132Bによる処理は、エージェント解号ステップ1428-I(図52)からエージェント配送ステップ1430-Iに移行する。エージェント配送ステップ1430-Iにおいて、エージェント150Aは、エージェント配送ステップ1422(図40)について前述したようにフローチャート1422(図49)に従って配送される。エージェント配送ステップ1430-Iのコンテキストにおいて、チケットは、今や実際のチケット1306(図35)であり、仮定されたチケットではない。エージェント配送ステップ1430-I(図52)において、エージェント150A(図50)は、プレイス220Bにおける占有が許容されており、エージェント150Aによって遂行されるオペレーション"go"は成功のうちに終了する。

【0614】処理はエージェント配送ステップ1430-I(図52)から終了ステップ1426-Iに移行し、このステップにおいて、フローチャート1400-I(図52)に従った処理、すなわちエンジン132Bによるシステムオペレーション"transferIn"の遂行が成功のうちに終了する。

【0615】フローチャート1400-I(図52)の各ステップを遂行する間に投出された例外は、フローチャート1450-I(図54)のステップの遂行を生じさせる。エージェント150Aは、ステップ1452-I(図52)において符号化エージェント150A-E(図53)から解号される。エージェント150Aはエンジン132B(図53)に到着しているので、エー

エージェント150Aは、エンジン132Aによって前述したように継続中のトランスファーのリスト中には保持されていない。

【0616】従って、エージェント150Aの実行を継続するためには、エージェント150Aを解号することが必要になる。処理はステップ1452-I（図54）からエージェント配送ステップ1454-Iに移行し、このステップにおいてエージェント150Aは、ステップ1426（図40）について前述したように、パーガトリに配送される。処理はエージェント配送ステップ1454-I（図54）から終了ステップ1456-Iに移行し、ここでフローチャート1450-Iによる処理が終了する。エージェント配送ステップ1454-Iの結果としてパーガトリを占有しているエージェント150Aは、例外を投出し、それによってフローチャート1450-Iが遂行されるため、エージェント150Aによって遂行されたオペレーション”go”は失敗となる。

【0617】従ってエージェント150A（図34、図41、図50、図51、図53）は、オペレーション”go”を遂行することによって、エージェント150Aとエージェント150Aによって所有されるオブジェクト例えばオブジェクト140A、140Bのネットワーク1500を通る移動すなわち転送を指示する。オペレーション”go”の遂行後に、このオペレーション”go”の直後のエージェント150A内の指令によって150Aの解釈が続けられる。

【0618】図36はエージェント150Aによるオペレーション”go”の遂行の直後のエージェント150Aの内部状態の実行状態の一部を示している。オペレーション”go”の遂行による結果であるチケットスタブ1308はスタック1304の上部にある。プレイス220Bはエージェント150Aが占有するプレイスを特定化するエージェント150Aのプロパティであり、それによってエージェント150Aがプレイス220Bを占有することを指示する。

【0619】イングレス (Ingress) 及びエグレス (Egress) : オペレーション”entering” 及びオペレーション”exiting”

前述したようにプレイス220Bは、オペレーション”entering”の遂行によってエージェント150Aへのイングレスを許容したり拒絶したりする。

【0620】プレイスをプレイス220Bが遂行することをリクエストする前に、符号化されたエージェント150A-E（図34、図41、図50、図51、図53）をエンジン132Bに配送する。本発明の別の実施例によれば、プレイス220Bは、エージェント150Aが通信リンク102AZを横切って通信下部構造132Z-CIを経て、更に通信リンク102ZBを横切ってエンジン132Bに転送される前に、オペレーシ

ョン”entering”を遂行するように指示される。

【0621】プレイス220Bは、オペレーション”entering”を遂行することによって、プレイス220Bを占有する許可をエージェント150Aに与えたり拒絶したりする。しかし、本発明は高度の潜在力とともに、広い領域のネットワークにおいて使用されるものと想定されている。

【0622】”待ち時間”（latency）という用語は、この明細書では、この技術で通常使用されているように、情報の送出者によって情報が最初に送出された時点と情報の受領者によって情報が最初に受領された時点との間の時間量を表すように用いられている。高度の待ち時間を持ったネットワークの場合、短いメッセージでも、目的点に到達するまでに実質的な量の時間を必要とすることがある。この場合プレイス220B（図34）がオペレーション”entering”を遂行することをリクエストして結果を待つことは、オペレーション”entering”を遂行することのリクエストを通信リンク102AZを横切って、通信下部構造132Z-CIを通り、通信リンク102ZBを横切り、さらにエンジン132Bを経てプレイス220Bに送出し、逆の経路を経てオペレーション”entering”によって生じた結果の受領を待つことを包含する。エンジン132Bにエージェント150Aを転送する前にこの動作を行うと、エンジン132A、132Bの間の待ち時間の2倍にほぼ等しい時間量だけエージェント150Aをエンジン132Bに転送する時間が遅延される。従って本発明の性能を実質的に向上させるためには、エージェント150Aをエンジン132Bに転送するまでプレイス220Bによるオペレーション”entering”の遂行を遅延させる。

【0623】エージェント150A（図50）がオペレーション”go”の結果として、プレイス220Bにエンタリングするコンテキストにおいて、オペレーション”entering”を以上に説明したが、オペレーション”entering”は、プレイス220B中においてプロセスが作りだされることの結果としてプロセス（エージェント又はプレイス）がプレイス220Bにエンタリングする結果としても実行される。プロセスの生成についてはアペンディクスAに詳細に説明されている。

【0624】図56は、オペレーション”entering”の遂行の直前のエージェント150Aの実行状態を表している。プロセス例えばエージェント150Aの実行状態を以下に詳細に説明する。オペレーション”entering”は、エンジン132Bのリクエストによって遂行され、オペレーション”entering”の遂行は、エージェント150A（図50）の実行状態中に記録される。エージェント150A又はプレイス220B、好ましくはプレイス220Bの許可は、オペレーション”entering”の遂行によって受け入れられ得る。

【0625】フレーム2200は、エージェント150

Aの実行状態の一部であり、プレイス220Bによって遂行されるオペレーション“entering”の動的な状態を記録する。フレーム2200は、現在のスタックであるスタック2202を含む。スタック2202は頂部から低部にかけて、接点2208、パーミット2206及びチケット2204を含む。

【0626】接点2208は、プレイス220Bに入ろうとするプロセスとしてエージェント150Aを特定化する。以下に、及びアペンディクスAに、詳細に説明するように、接点は、そのプロパティの中に、プロパティ“subjectName”及びプロパティ“subject”を有する。接点2208（通常はエージェント150Aのリファレンスである）のプロパティ“subject”は、オペレーション“entering”においては0であるため、プレイス220Bがエージェント150AにインGRESを許可する前は、プレイス220Bにはエージェント150Aの参照（リファレンス）が与えられない。接点2208のプロパティ“subjectName”は、プレイス220BへのインGRESをリクエストするエージェントとしてエージェント150Aを特定化するテレネームである。

【0627】パーミット2206は、プレイス220Bにエンター、すなわち入ろうとするエージェントの提案されたローカル許可である。パーミット2206は、“byProtectedRef”を通過しており、従ってオペレーション“entering”の遂行によっては変更されない。アーギュメント及び結果の通過、特に保護されたリファレンス例えば“byProtectedRef”による通過は以下に詳細に説明されている。エージェントのローカル許可は、与えられたプレイスにある間エージェントの能力を規定する。一例としてエージェント150Aは、能力及びアローワンスのサブセットを規定するパーミット2206をプレイス220Bに供給することによって、プレイス220Bにある間に、そのネイティブパラミット（許可）の能力及びアローワンスのサブセットにそれ自身を制限する。

【0628】チケット2204は、“byProtectedRef”を通過し、回答するプレイス220Bに移動するために使用されるチケットと同等であり、エージェント150Aが別のプレイスからエンタリングしようとしていることを回答するプレイス220Bに通報する。チケット2204が0である場合、プロセスは、回答するプレイス220B内においてプロセスが生成されることによって回答プレイス22000Bにエンタリングするように努める。

【0629】あるプレイスを容認するか否かを定める方法、すなわちクラス“Place”によって規定されたオペレーション“entering”をインプリメントする方法は、クラス“占有拒絶”のメンバである例外を常に導出することによって、回答プレイスへのプロセスのインGRESを拒絶する。しかし、本発明のユーザーは、特定の条件

のもとにプロセスへのエンタリングを許可するクラス“Place”のサブクラスを規定する。エージェントが1のプレイスから別のプレイスへ移動するためにはこれは実際には必要なことである。

【0630】オペレーション“entering”（1つのプレイスによって実行される）のインプリメンテーションは、そのプレイスが1つのインスタンス（例）であるクラスによって供給されるか、又は受け継がれる特別な方法によって規定されているので、各々のプレイスは、ネットワーク中の他のプレイスと異なった仕方、オペレーション“entering”をインプリメントすることができる。論理フローダイアグラム2260（図55）は、オペレーション“entering”のインプリメンテーションの一例として、従って、ステップ1448（図40）に入る際のオペレーション“entering”の遂行においてプレイス220Bによって取られる各ステップのインプリメンテーションの一例として役立つ。

【0631】接点2208、パーミット2206及びチケット2204は、それぞれステップ2262、2264及び2266において、スタック2202から復元（ポップ）される（図55）。処理は次にチケットテストステップ2272に移行し、ここでチケット2204が0と比較される。チケット2204が0であると、エージェント150Aは局地的に作りだされ、処理は終了ステップ2274に移行し、ここでオペレーション“entering”は成功のうち終了する。しかし、ある状態のもとにオペレーション“entering”が失敗し、プレイス内のプロセスの生成が阻止されるように、プレイスを規定することができることは、以上の説明の視点及び以下の説明から明らかである。

【0632】チケット2204が0でない場合、処理はチケットテストステップ2272からアローワンステストステップ2276に移行する。アローワンステストステップ2276においては、パーミット2206のプロパティ“charge”が、パーミット2206のアトリビュート“charge”を質疑することによって生成され、パーミット2206のプロパティ“charge”は1000と比較される。パーミット2206のプロパティ“charge”が1000より多い場合、序理はアローアエンステストステップ2276から終了ステップ2278に移行し、そこでクラス“占有拒絶”の例外が投出され、回答プレイスへのエージェントのインGRESが拒絶される。パーミット2206のプロパティ“charge”が1000に等しいかそれよりも小さい場合、処理はアローアエンステストステップ2276から終了ステップ2270に移行し、そこでオペレーション“entering”は成功のうち終了し、エージェントは回答プレイスにインGRESすることが許可される。図55の実施例において最大の許可可能なチャージアローアエンスとしての正確な数値1000は、説明の目的のために任意に選ばれたものにすぎ

ない。

【0633】論理フローダイアグラム2260の方法は、リソースが限定されているパーソナルコンピュータシステム内において動作するエンジンによって処理されるプレイスのためのオペレーション“entering”の適切なインプリメンテーションの例示を示している。以下に、またアペンディクスAに説明するように、許可のプロパティ“charge”は、処理の許可である。従ってその提案されたローカルパーミットにおいて大きなチャージアロワンスによって大量の計算及びそのリソースの必要を指示するエージェントは、論理フローダイアグラム2260によってイングレスが拒絶される。従って小型のパーソナルコンピュータシステムのオーナーは、ローカルに作りだされたプロセス及び他の場所で作りだされたエージェントによる比較的廉価な訪問に対してそのパーソナルコンピュータシステムを制限することができる。

【0634】プレイス220B（図57）によるオペレーション“entering”の遂行直後には、オペレーション“entering”がいかなる結果も生じなかったので、スタック2200にはエンpty（空）である。

【0635】前述したように、あるプレイスは、離去をノートする。すなわちオペレーション“exiting”の実行によるプロセスの占有の終了をノートする。論理フローダイアグラム1400（図37、図40）のコンテキストにおいて、プレイス220Aは、エージェントすなわちエージェント150Aがオペレーション“exiting”の結果としてプレイス220Aを離去することの結果として、オペレーション“go”を実行している。オペレーション“exiting”は、あるプロセス（エージェント又はエージェント）がプロセスの破壊の結果としてもはやプレイス220Aを占有していない場合にも、プロセス220Aによって遂行される。プロセスの破壊はアペンディクスAにより詳細に説明されている。クラス“Place”によって規定されたオペレーション“exiting”のインタフェイスは、図58及び図59に示されている。

【0636】オペレーション“exiting”は、エンジン132A（図34）のリクエストによって遂行されるので、オペレーション“exiting”の動的状態を記録しているフレーム2300（図58）は、プレイス220A（図34）の実行状態の一部でもなければ、エージェント150Aの実行状態の一部でもない。実際にプレイス220Aがオペレーション“exiting”を遂行している時点においては、エージェント150Aはもはやプレイス220Aを占有しない。エンジン132Aは、“エンジンプロセス”の一部として新しい実行状態を作りだす。エンジンプロセスは、オペレーション“exiting”又はオペレーション“parting”の遂行の目的のためにエンジンによって作りだされたプロセスである。オペレーション“parting”は、以下及びアペンディクスAに

詳細に説明されている。

【0637】図58は、オペレーション“exiting”の遂行の直前のフレーム2300を示している。フレーム2300は現在のスタックである、スタック2302を含む。スタック2302は、上部から下部にかけて、オペレーション“exiting”のアーギュメントとして、接点2308、パーミット1306及びチケット2304を備えている。

【0638】接点2308は、プレイス220Aをエクシットするプロセスとして、エージェント150A（図34）を特定する。以下にまたアペンディクスに詳細に説明するように、コンタクトは、そのプロパティの中に、プロパティ“subjectName”及びプロパティ“subject”を有している。接点2308（図58）のプロパティ“subjectName”は、エージェント150A（図34）、接点2308（図58）のプロパティ“subjectName”は、エージェント150A（図34）を特定化する。通常エージェント150A（図34）への参照であるコンタクト230（図58）のプロパティ“subject”は、オペレーション“exiting”においてボイドされるので、プレイス220Aはエクシットするエージェント150Aへの参照とともに残されない。

【0639】パーミット2306（図58）は回答プレイス220A（図34）すなわちエージェント150Aをエクシットする現在のローカルパーミットである。プロセスのローカルパーミットは、ある与えられたプレイスを占有している間のプロセスの能力を限定する。ローカルパーミットはアペンディクスAに詳細に説明されている。パーミット2306（図58）は、“パイプロテクディットレフ”を通過しているので、オペレーション“exiting”を遂行する際に、プレイス220A（図34）によって変更されることはできない。

【0640】エクシットするプロセスのローカルパーミットすなわちパーミット2306（図58）は、プレイス220Aの占有の間エクシットプロセスによって使用されていたリソースの形式及び量を定めるために、回答プレイス220A（図34）の解釈をするエンジンによって使用される。そのため、エクシットするプロセスのオーソリティーは、それに含まれるリソース及びプレイス220Aについて使用するためにビル（BILL）することができる。使用されるリソースの形式及び量は、パーミット2306（図58）とエクシットするプロセスへのイングレスを許可する際に、オペレーション“entering”においてアーギュメントとして使用された許可との比較によって定められる。オペレーション“entering”は、以上にまたアペンディクスAに詳細に説明されている。

【0641】一例としてプレイス220A（図34）は、オペレーション“entering”を遂行する際に、オペレーション“exiting”の遂行において消費されたパー

ミット2306（図58）と後に比較するために、提案されたローカル許可、例えばパーミット2206（図56）を格納しておくことができる。以下にまたアペンディクスAに詳細に説明されるように、接点例えば接点2308は、プロパティ“subjectノーツ”を有する。一実施例によれば、プレイス220Aは、コンタクト2208（図56）のプロパティ“サブジェクトノーツ”内に、オペレーション“entering”を遂行する際のパーミット2206を格納している。オペレーション“exiting”を遂行する際に、プレイス220A（図34）は、接点258のプロパティ“サブジェクトノーツ”中に格納された比較2206（図56）をオペレーション“exiting”の遂行において消費されたパーミット2308（図58）比較することによって、プレイス220Aを占有している間にエージェント150Aが消費するリソースの量を定める。

【0642】チケット2304（図58）は、通過した“byProtectedRef”であり、チケット1306（図35）に等しい。このチケットは、オペレーション“go”を遂行する結果としてエージェント150Aが回答プレイス（リスポンディングプレイス）220Aを離去していることを回答プレイス220Aに通知する。チケット2304（図58）が0である場合、エージェント150A（図34）は、エージェント150Aの破壊の結果として回答プレイス220Aをエクシットする。

【0643】フレーム2300（図59）は、プレイス220A（図34）によるオペレーション“exiting”の遂行の直後に示される。スタック2302（図58）は、オペレーション“exiting”がいかなる結果も生じなかったため、エンプティである。フレーム2300（図59）は、プレイス220A（図34）又はエージェント150Aの実行状態の一部分ではなくその代わりにエンジンプロセスの実行状態の一部であるため、オペレーション“exiting”の遂行によって投出されたいかなる例外も、プレイス220A又はエージェント150Aによって経験されず、従ってプレイス220A又はエージェント150Aにいかなる影響も持たない。プレイス220Aは、オペレーション“exiting”を遂行する際に、エージェント150A及び、エージェント150Aによって所有されるオブジェクト140A、140Bの離去をノート（検知）する。

【0644】オブジェクトインタチェンジ1つのプレイスから別のプレイスへエージェントを転送するために必要とされる時間は、エージェントに含まれるオブジェクトの転送を目的プレイスに同等のオブジェクトと思われるオブジェクトのみに制限することによって、実質的に減少する。エンジン132Aから132Zに移動するエージェント150Aが、エージェント150A及びエージェント150Aの所有するオブジェクトがメンバとなっているクラスを表すクラスオブジェクトを含むため、

これらのオブジェクトの転送の制限は特に重要である。クラスオブジェクトは通常は非常に大きく、そして1つのエージェント及びそのエージェントが所有するオブジェクトがメンバとなってクラスは典型的には非常に多数あるので、これらの全てのクラスオブジェクトを転送することは実用的でない。従って、エンジン132Aから132B（図34）にエージェント150Aを転送する際に、エンジン132B中のクラスオブジェクトによって表されないクラスを表すクラスオブジェクトのみをエージェント150Aと共にエンジン132Bに転送することが望ましい。

【0645】ネットワーク1500（図34、図41、図50、図51、図53）中の多くのプレイスにおいて同等のオブジェクトを持つと思われるオブジェクトは、ニックスインクラス“インタチェンジド”のメンバであり、従って、プロパティ“ダイジェスト”を有する。ミックスインクラス“インタチェンジド”のメンバは、“インタチェンジドオブジェクト”と呼ばれる。インタチェンジされたオブジェクトのプロパティ“ダイジェスト”であるオブジェクトは、インタチェンジされたオブジェクトの“ダイジェスト”とも呼ばれる。

【0646】インタチェンジされたオブジェクトは、エンジン例えばエンジン132A又は132B（図34）が、インタチェンジされたオブジェクトのダイジェストに等しいダイジェストを持つインタチェンジされたオブジェクトのクラスの他の任意のインスタンスと同等であるとみなしたオブジェクトである。ダイジェストは、全ての他のものから第1のインタチェンジドオブジェクトを区別する目的に適したオブジェクトである。一例として、インタチェンジされたオブジェクトの標準的な2進表示の数学的ハッシュは、インタチェンジドされたオブジェクトの多くのクラスに適している。クラスオブジェクトすなわちクラス“class”のオブジェクトは、以下に、及びアペンディクスAに、詳細に説明するように、作り付けられたクラス、及びユーザーによって規定されたクラスを規定し、コンピュータネットワーク1500内のエンジン132A間において変動しない。そのためクラスオブジェクトは、インタチェンジされたオブジェクトである。

【0647】移送されるデータの量、従ってエージェントの転送に要する時間、及び1つのプレイスから次のプレイスにそのエージェントによって所有されるオブジェクトを転送するための時間を減少させるダイジェストを使用することは、論理フローダイヤグラム2400及び2450（図60及び図61）に示されている。

【0648】論理フローダイヤグラム2400（図60）は、インタチェンジ（相互変換）されたオブジェクトを検討する場合に、エージェント150A（図34）及びエージェント150Aが所有するオブジェクトをソースエンジン132Aから目的エンジン132Bに転送

するプロセスを表している。あるエージェントは、そのエージェントによって所有される全てのオブジェクト、そのエージェントがメンバとなっている各々のクラス、及びそのエージェントが所有するオブジェクトがメンバとなっている各々のクラスを”包含”する。前述したように、エージェント150A及びエージェント150Aによって包含される全てのオブジェクトは、 SHIPPINGボックスにパッケージされ、符号化され、エンジン132Bに転送される。エージェント150A及びそれに包含されるオブジェクトをパッケージにする場合、そのエンジン132Aは、ステップ2402（図60）においてエージェント150Aが包含する全てのオブジェクトを収納している。処理はステップ242からwherein each objectステップ2404に移行する。このステップ2404は、次のステップ1412とともに、ステップ2402において収納された各々のオブジェクトが検討される1つのループを規定する。処理はfor each objectステップ2404からダイジェストテストステップ2406に移行し、そのステップ2406において、エージェント150Aが包含する各々のオブジェクトのクラスのメンバ構成がチェックされ、それによって各々のオブジェクトがダイジェストを持つか否かが定められる。

【0649】前述したようにまたアペンディクスA及びEに一層詳細に示すように、ミックスインクラス”インタチェンジド”のメンバは、ダイジェストを含むことができる。従って、オブジェクトがダイジェストを有する場合、そのオブジェクトはインタチェンジされたオブジェクトである。インタチェンジされたオブジェクトがダイジェストを持たない場合すなわちインタチェンジされたオブジェクトのアトリビュート”ダイジェスト”が0である場合、そのインタチェンジされたオブジェクトは、インタチェンジされているのではなくて、その代わりに、インタチェンジされていないオブジェクトとして取り扱われる。

【0650】オブジェクトがダイジェストを持たない場合には、そのオブジェクトがインタチェンジされたオブジェクトであっても、処理はダイジェストテストステップ2406からステップ2410に移行し、このステップ2410において、オブジェクトが、アペンディクスBの符号化規則にしたがって符号化され、SHIPPINGボックスに収納される。さもなければ、そのオブジェクトがダイジェストを有する場合、処理はダイジェスト2406からステップ2408に移行し、このステップ2408において、そのオブジェクトのダイジェストとそのオブジェクトが1つのインスタンスであるそのクラスのサイテーションとが、アペンディクスBの符号化規則によって符号化され、部品ボックスに収納され、その部品ボックスがSHIPPINGボックスに収納される。

【0651】処理はステップ2408又は2410から、次のステップ2412に移行し、次のステップ24

12において、for each object ステップ2404に移行する。エンジン132Aは、for each object ステップ2404と次のステップ2412とに規定されるループにおいて、エージェント150Aに包含される各々のオブジェクトについて、ステップ2406、2408、2410のプロセスを反復する。ステップ2402において収納された全てのオブジェクトが、for each object ステップ2404及び次のオブジェクトステップ2412のループにしたがって処理されたならば、処理はfor each objectステップ2404からtransfer out encoded agent ステップ2414に移行する。

【0652】transfer out encoded agent ステップ2414において、SHIPPINGボックスは符号化され、ソースエンジン132Aによって目的エンジン132Bに、前述したように転送される。従って、目的エンジンに転送される符号化されたエージェントは、ダイジェストを持たないエージェントによって包含される全てのオブジェクト、及びエージェントによって包含される相互変換可能でダイジェストを有する全てのオブジェクトのダイジェストを包含している。transfer out encoded agent ステップ2414は、ネットワークを横断するエンジン間の相互作用を示すために、二重ボックスによって表されている。エンジン間のエンジンデータのこのような交換は、以上の説明した通りであり、またアペンディクスC及びFに記載されている。

【0653】論理フローダイアグラム2450（図61）は、目的エンジン132Bのパーспекティブからの符号化されたエージェント150Aのトランスファを示している。目的エンジン132Bは、transfer out encoded agent ステップ2452において、ソースエンジン132Aから2進データとして符号化エージェントを受ける。図34、図41、図50、図51、図53のコンテキストにおいて前述したように、符号化されたエージェントは、エンジン132Aからエンジン132Bに至る途中の、1以上の中間のエンジン例えばエンジン132Bを通過することができる。

【0654】処理はトランスファーアウトエンコードエージェントステップ2452からステップ2454に移行し、このステップ2454において、符号化エージェントは復号され、それにより、そのエージェントによって包含されるエージェントオブジェクトを内容物とするSHIPPINGボックスが再構成される。ダイジェストを持たず、エージェントによって包含される各々のオブジェクトと、ダイジェストを有するエージェントによって包含されるインタチェンジされたオブジェクトのダイジェストとは、for each object ステップ2456及び次のステップ2462によって形成されるループにしたがって、目的エンジン132Bによって消費される。このループの各々の反復は、ステップ2454において改号されたオブジェクトの一つを処理する。



【0655】処理はfor each object ステップ2454からダイジェストテストステップ2464に移行する。ダイジェストテストステップ2464において目的エンジン132Bは、あるオブジェクトがダイジェストであるか否かを定める。そのオブジェクトがダイジェストでない場合、処理はダイジェストテストステップ2464から直接に、次のオブジェクトステップ2462に移行する。逆にそのオブジェクトがダイジェストであれば、処理はダイジェストテストステップ2464からテストステップ2466に移行し、そのオブジェクトは、“サブジェクトダイジェスト”と呼ばれる。テストステップ2466において、目的エンジン132Bは、サブジェクトダイジェストがエンジン132Bに存在するインタチェンジされたオブジェクトと同等であるか否かを定める。このような決定がなされる方法については、以下に詳細に説明する。

【0656】同等のダイジェストを有するインタチェンジされたオブジェクトがステップ2466において見出されると、処理は、テストステップ2466から代替ステップ2468に移行する。代替ステップ2468において、インタチェンジされたオブジェクトは、見出されたインタチェンジされたオブジェクトに関してサブジェクトダイジェストによって表されるインタチェンジされたオブジェクトに、ステップ2454において復号されるオブジェクト内の全てのリファレンス（参照）を代替することによって、サブジェクトダイジェストに代替される。処理は代替ステップ2468から次のオブジェクトステップ2462に移行する。

【0657】同等のダイジェストを有するインタチェンジされたオブジェクトがテストステップ2466において見出されなかった場合、処理はテストステップ2466からステップ2470に移行する。ステップ2470では、サブジェクトダイジェストがダイジェストのリストに付加される。新しいリストが存在しない場合には新しいリストが作りだされる。処理はステップ2470から次のステップ2462に移行する。

【0658】処理は、次のステップ2462からfor each object ステップ2456に移行する。ステップ2454において復号された全てのオブジェクトがfor each object ステップ2456と次のステップ2462とのループにしたがって処理された場合には、処理はfor each object ステップ2456から、以下に説明するリストテストステップ2476に移行する。

【0659】エンジンはステップ2466において次のようにして相互変換可能な（インタチェンジ可能な）オブジェクトをサーチする。これらのエンジンは、同等のインタチェンジ可能なオブジェクトを交換することのできる1以上のプレイスを備えている。各々のこのようなプレイスは、そのプレイス内に存在するインタチェンジ可能なオブジェクトの“デポジトリ”を備えている。デ

イクショナリ2490（図62）はプレイス220B

（図53）のデポジトリである。ディクショナリ2490（図62）のキーは、クラス、すなわちクラス2400K1、2400K2、2400K3及び2490K4である。各々のクラスにはディクショナリが組み合わされている。例えばクラス2490K1-2490K4には、ディクショナリ2490A、2490B、2490C及び2490Dがそれぞれ組み合わされている。

【0660】各々のディクショナリ2490A-2490Dは、同一の一般的な組織を有している。ディクショナリ2490Aは例示的である。ディクショナリ2490Aのキーはダイジェスト、すなわちダイジェスト2490AK1、2490AK2及び2490AK3である。各々のダイジェストには、ダイジェストが組み合わされたダイジェストであるインタチェンジされたオブジェクトが組み合わされている。一例として、ダイジェスト2490AK1-2490AK3には、インタチェンジされたオブジェクト2490AV1、2490AV2及び2490AV3がそれぞれ組み合わされている。

【0661】インタチェンジされたオブジェクトは、2ステップによってディクショナリ2490から検索される。前述したようにダイジェストとあるプラスのサイテーションとは、インタチェンジされたオブジェクトを表すために、部品ボックスによって使用される。第1のステップにおいて、サイテーションを使用して、そのサイテーションによって参照されたクラスに組み合わされたディクショナリ2490A-2490Dの内の1つのディクショナリを検索する。一例として、サイテーションがクラス2490K1を参照した場合、ディクショナリ2490は検索される。

【0662】第2のステップでは、部品ボックスに含まれるダイジェストと同等のダイジェストとインタチェンジされたオブジェクトとの間の関連について、検索されたディクショナリがサーチされる。その関連が乱されたら、インタチェンジされたオブジェクトは検索される。関連が乱されなかった場合、そのプレイスは、部品ボックス中のダイジェスト及びサイテーションによって表されたインタチェンジされたオブジェクトと同等のインタチェンジされたオブジェクトを備えていない。従って、ディクショナリ2490（図62）のようなデポジトリは、同等の相互変換（インタチェンジされたオブジェクト）をあるプレイスを含むか否かを定め、そのようなオブジェクトを検索するに用いられる。

【0663】前述したように、処理はfor each object ステップ2456からリストテストステップ2476に移行する。リストテストステップ2476においては、目的エンジン例えばエンジン132Bは、ダイジェストのリストがステップ2470において作り出されたか否かを定め、作り出された場合には、リストがエンブティであるかどうかを定める。ダイジェストのリストが存在

しないかまたはエンプティである場合、処理はリストテストステップ2476から、activate traveling agentステップ2478に移行し、このステップ2478においては、エンジン132Bには、エージェント150Aを実行についてスケジューリングすることによって、移動中のエージェント150Aを活性化する。処理は、activate traveling agentステップ2478から、終了ステップ2479に移行し、この終了ステップにおいて、エンジン132Bによってなされたエージェント150Aのトランスファーが成功のうちに終了する。

【0664】もし、ステップ2470においてリストが作り出され、そのリストが少なくとも1つのダイジェストを含んでいた場合、目的エンジン132Bには、少なくとも1つのインターチェンジ（相互変換）されたオブジェクトのための同等のオブジェクトが見い出されず、処理は、リストテストステップ2476から、hold traveling agentステップ2471に移行する。hold traveling agentステップ2471において、移動中のエージェント150Aは、中断された状態において、エンジン132B内の保持列中に保持される。処理は、hold traveling agentステップ2471からステップ2472に移行する。

【0665】ステップ2472において、目的エンジン例えばエンジン132Bは、オブジェクト開始エージェントすなわち“ORA”を作り出す。ORAは、ソースエンジン例えばエンジン132Aから、フローチャート2480（図63）に従って、インターチェンジされたオブジェクトを検索する。処理はステップ2472からステップ2474に移行し、そこでORAは、オペレーション“live”を遂行するように指示され、それによりORAの解釈が開始される。以下に詳細に説明するように、ORAの中心活動性は、フローチャート2480

（図63）によって示される。処理はステップ2474から終了ステップ2479に移行し、ここでエージェント150Aのトランスファー（エンジン132Bによって行なわれる）が成功のうちに終了する。尚、移行中のエージェント150Aは、活性化されず、 SHIPPINGボックスの形状にある。エージェント150Aは、SHIPPINGボックスから再構成され、以下に説明するように、ORAによって再活性化される。

【0666】フローチャート2480（図63）は、ORAの中心プロシージャを表している。ステップ2482において、ORAは、ソースエンジン例えばエンジン132Aに、オペレーション“go”の遂行によって移動する。処理は、ステップ2482からステップ2484に移行し、このステップ2484において、ORAは、エンジン132A中のオブジェクトレポジトリ例えばディレクショナルリ2490（図62）から、ダイジェ

ストのリストに含まれるダイジェストと同等のダイジェストを有する各々のオブジェクトのコピーを収集する。処理はステップ2484からステップ2485に移行する。

【0667】ステップ2485において、ORAは、目的エンジン例えばエンジン132Bに、オペレーション“go”の遂行によって移動する。ORAは、ダイジェストのリストにそのダイジェストが含まれないオブジェクトのコピー、すなわち、目的エンジン中に同等のオブジェクトが存在しないオブジェクトのコピーを、目的エンジンに持参する。処理はステップ2485からステップ2486に移行する。ステップ2486において、ORAは、収集されたオブジェクトを、以下に説明するように、エージェント150A内のダイジェストと代替する。処理はステップ2486からactivate traveling agentステップ2487に移行し、このステップ2487においてエージェント150Aは、SHIPPINGボックスから再構成され、前述したようにして活性化される。エージェント及びそのエージェントに包含されるオブジェクトをSHIPPINGボックスから再構成することは、アペンディックスDに詳細に説明されている。処理はステップ2487から終了ステップ2488に移行し、この終了ステップにおいてORAの中心プロシージャが成功のうちに終了する。

【0668】このように、目的エンジン中のインターチェンジされたオブジェクトと同等でないそのインターチェンジされたオブジェクトのみを、ネットワーク通信メディアを横切って転送することによって効率化が達成される。

#### 【0669】オペレーション“send”

あるエージェントは、オペレーション“send”を遂行することによって同時にいくつかのプレイスに移行することができる。図64は、エージェント150Aによるオペレーション“send”の遂行前のネットワーク2500の状態を示している。図64の例において、エンジン132Aのデータ部分132A-D中のエージェント150Aは、エンジン132Bのデータ部分132B-D中のプレイス220にそれ自身のクローンを転送しまたそれと同時にエンジン132Cのデータ部分132C-D中のプレイス220Cにそれ自身のクローンをそれぞれ転送するように構成されている。換言すれば、エージェント150Aは、エージェント150Aのクローンのための目的プレイスとしてのプレイス220B、220Cを特定化する2つのチケットのリストをオペレーション“send”へのアーギュメントとして供給して、オペレーション“send”を遂行する。エージェント150Aのクローンは、エージェント150Aのコピーであり、このコピーは、エージェント150Aの実行状態を含めてエージェント150Aを複製してコピーを作成することによって形成され、そのコピーに新しい名前と識別子と

を割り当ててエンジンによる実行のためにそのコピーをスケジューリングすることによって”活性化”されたものである。クローンについてはアペンディックスAにおいて一層詳細に説明する。

【0670】オペレーション”send”のインタフェースは、図65、図66及び図67に示されている。図65はフレーム2602を示し、このフレーム2602は、エージェント150Aによって実行されたオペレーション”send”の動的状態を記録し、オペレーション”send”の遂行の直前のエージェント150Aの実行状態の一部分である。エージェントを含めたプロセスの実行状態は以下にまたアペンディックスA及びBに詳細に説明されている。

【0671】スタック2604はフレーム2602に含まれている。スタック2604は、アーギュメントがそれから復元（ポップ）され、オペレーション”send”の遂行の間に結果がプッシュ（保存）されるスタックである。”現在のスタック”は、以下にそしてアペンディックスA、Bに一層詳細に規定され説明される。

【0672】大文字”T”によって示されたスタック2604の上部には、チケット2608及び2610をアイテムとするリスト2606がある。リスト2606のチケット2608及び2610の各々は、エージェント150Aの対応するクローンが行なうトリップを各々定義している。オブジェクト2618及び2620は、それぞれチケット2608、2610のプロパティ”travelNote”である。オブジェクト2618、2620は、オペレーション”send”の遂行によって作り出されたそれぞれのクローンを識別可能とするためのメカニズムを提供する。一例として、チケット2608、2610を形成する場合に、エージェント150Aは、それぞれのテキストが”Able”及び”Baker”であるストリングをオブジェクト2618及び2620中に格納することができる。このような場合に、チケット2608に対応するエージェント150Aのクローンは、チケット2608のプロパティ”travelNote”からストリング”Able”を検索する。同様に、チケット2610に対応するエージェント150Aのクローンは、チケット2610の”travelNote”からストリング”Baker”を検索する。

【0673】それぞれのクローンが識別される特別のインプリメンテーションは、最終的には、本発明のユーザーの責任である。以下は別の例である。オブジェクト2618中には、その値が1である整数があり、それによって、リスト2606中の対応のチケット2608の位置を表している。同様に、オブジェクト2620中には、その値が2である整数が格納されており、それによってリスト2606中の対応するチケット2610の位置を表している。

【0674】パーミット2622、2624は、それぞ

れチケット2608、2610のプロパティ”destinationPermit”である。インテジャー2626、2628は、それぞれパーミット2622、2624のプロパティ”charges”である。

【0675】パーミット2612は、エージェント150Aのプロパティ”permit”であり、従って、エージェント150Aの内部状態の一部を表している。以上に説明したアペンディックスAに説明するように、パーミット2612は、エージェント150Aの実行を制限する。ブーリアン2614はパーミット2612のプロパティ”canSend”である。ブーリアン2614の値が”false”であれば、オペレーション”send”は失敗し、クラス”Permit Violated”の例外を投出する。パーミット2612のプロパティ”charges”は整数2616である。アペンディックスAに一層詳細に説明するように、整数2616は、エージェント150Aのパーミット処理のアローワンスを表している。

【0676】図64の例において、その現在のロケーションがプレイス220Aであるエージェント150Aは、エージェント150Aのそれぞれのクローンをプレイス220B、220Cに転送するように形成されている。エージェント150は、リスト2606（図65）のチケットによって規定された各々の目的プレイスにそれ自身のクローンを送出するので、オペレーション”send”の遂行は、時には、リスト2606中のチケットの数と同数のエージェント150Aのクローンを作り出す。図68、図69は、オペレーション”send”の簡単な実施例を表し、ここにエンジン132A（図68）は、リスト2606（図65）中のチケットと同数のエージェント150Aのクローンをデータ部分132A-D中に作り出す。この例では、リスト2606（図65）は、2つのチケット、すなわちチケット2608、2610を備えている。従って、エンジン132Aは、エージェント150A（図69）の2つのクローンすなわちエージェント150A-1、エージェント150A-2を作り出す。応答エージェントのクローニングが延期されるより効率的な実施例については以下に説明する。エージェント150Aのクローンを形成する場合、パーミット2622、2624は、作り出されたクローンのプロパティ”permit”とされる。換言すれば、パーミット2622は、エージェント150A-1のプロパティ”permit”であり、パーミット2624は、エージェント150A-2のプロパティ”permit”である。エージェント150A-1、エージェント150A-2を作り出す際に、エージェント150Aすなわちクローンされているエージェントのパーミット2612の対応するアローワンスからパーミット2622、2624のアローワンスが差し引かれる。換言すれば、エージェント150A-1、150A-2が作り出された時にエージェント150Aのチャージのアローワンス、すなわちエ

ージェント150Aのパーミットのプロパティ”charges”である整数2616から整数2626、2628の値が引算される。整数2616が整数2626、2628の合計よりも大きくないか、又はこれに等しい場合には、オペレーション”send”は失敗し、クラス”Permit Violated”の例外が投出される。その理由は、エージェント150Aのチャージャローワンスがエージェント150Aから作り出されたクローンのチャージャローワンスの合計よりも小さいからである。

【0677】従って、オペレーション”send”のアーギュメントとして供給されたチケットの一部分であるパーミットは、オペレーション”send”の遂行により作り出されたクローンのネーティブパーミットを形成する。チケット2608、2610は、エージェント150Aのそれぞれのクローンが行なうトリップを規定し、パーミット2622、2624は、それぞれチケット2608、2610のプロパティ”permit”であり、従って応答（レスポnding）エージェント150Aのそれぞれのクローンのローカルパーミットである。

【0678】本発明の第2の実施例によれば、オペレーション”send”は、リスト2606に追加されるアーギュメントとして、整数（図示しない）を消費する。応答エージェント150Aのクローンを作り出す際に、それぞれのネーティブパーミットは、前述したように作り出されるが、各々のネーティブパーミットのプロパティ”charges”は最初は、消費される整数に等しい。

【0679】本発明の第3の実施例は、第2の実施例と同様であるが、例外として、整数のリストが単一の整数の代わりに消費される。さらに、リスト2606（図65）中のチケットのプロパティ”permit”から投出された各々のネーティブパーミットのプロパティ”charges”は、ネーティブパーミットがそれから導かれたチケットのリスト2606中の位置に等しい消費された整数リスト中の位置にある整数に最初は等しい。一例として、リスト260中の位置1にあるチケット2608に対応するクローンのネーティブパーミットのプロパティ”charges”は、消費された整数リスト（図示しない）中の1にある整数に最初は等しい。同様に、リスト2606中の位置2にあるチケット2610に対応するクローンのネーティブパーミットのプロパティ”charges”は、消費された整数のリスト中の位置2にある整数に最初は等しい。

【0680】本発明第4の実施例によれば、オペレーション”send”は、リスト2606に追加されるアーギュメントとして、パーミット（図示しない）を消費する。応答エージェント150Aのクローンを作り出す際に、各々のネーティブパーミットは、消費されるパーミットのコピーである。本発明の第5の実施例は前述した第4の実施例と同様であるが、パーミットのリストが単一のパーミットの代わりに消費される点で異なっている。さ

らに、リスト2606（図65）中のチケットに対応するそれぞれのクローンの各々のネーティブパーミットは、クローンが対応するチケットのリスト2606中の位置に等しいパーミットの消費されたリスト中の位置にあるパーミットに最初は等しい。一例として、リスト260中の1にあるチケット2608に対応するクローンのネーティブパーミットは、消費されたパーミットのリスト（図示しない）中の位置1にあるパーミットに最初は等しい。同様に、リスト260中の位置2にあるチケット2610に対応するクローンのネーティブパーミットは、消費されたパーミットのリスト中の位置2にあるパーミットに最初は等しい。

【0681】各々の前述した実施例において、応答エージェント150Aのパーミットであるパーミット2612のプロパティ”charges”はオペレーション”send”の遂行において作り出されたエージェントのクローンのネーティブパーミットのそれぞれのプロパティ”charges”の合計分だけ減少する。

【0682】エージェント150Aの各々のクローンは、目的プレイスとしての対応するチケットによって特定化されたプレイスに転送される。一例として、チケット2608は、エージェント150A-1の目的点としてプレイス220Bを特定化し、チケット2610は、エージェント150A-2の目的点としてプレイス220Cを特定化する。各々のクローンは対応する目的エンジン例えばエンジン132B又はエンジン132Cとのソースエンジン132Aの共働によってそれぞれの目的プレイスに移動する。オペレーション”send”を遂行するにあたってエージェントクローン例えばエージェント150A-1又はエージェント150A-2を1つのプレイスから次のプレイスに移送させることは、オペレーション”go”について以上に詳細に説明した通りである。

【0683】オペレーション”send”の遂行においてエージェント150Aがエージェント150A-1、150A-2を成功のうちに作り出した場合には、エージェントクローンのどれか1つ又は全部の転送が失敗したとしても、エージェント150Aによるオペレーション”send”の遂行は成功する。あるエージェントクローンのトリップが失敗した場合、最初にオペレーション”send”を遂行したエージェントではなく、エージェントクローンによって、トリップの例外が投出される。

【0684】エージェント150Aによるオペレーション”send”の遂行の直後のエージェント150Aの実行状態の一部は図66に示されている。前述した現在のスタックであるスタック2604は、ゼロオブジェクト2630を収容している。ゼロオブジェクトはもとのエージェントについて結果として生じ、それによってオペレーション”send”の遂行において作り出されたエージェントクローンからもとのエージェントを識別する。

【0685】図67は、エージェント150Aによるオペレーション”send”の遂行後のエージェント150A-1の実行状態の一部を示している。エージェント150Aによるオペレーション”send”の遂行後のエージェント150A-2の実行状態は、以下に説明するように、エージェント150A-1の実行状態に直接類似している。

【0686】パーミット2622は、エージェント150A-1のプロパティ”permit”である。スタック2604-1は、エージェント150A-1の生成において作り出されたスタック2604のコピーである。スタック2604-1はエージェント150A-1の現在のスタックである。スタック2604-1の上部にはチケットスタブ2632がある。チケットスタブ2632はオペレーション”send”の遂行によって作り出されたエージェントクロンのオペレーション”send”によって生じた結果である。チケットスタブ2632はチケット2608から導かれる。

【0687】チケットスタブ2632はクラス”チケットスタブ”のメンバである。チケットスタブ2632は他のプロパティの中でも、プロパティ”way”及び”travelNote”を有する。チケット2608はクラス”チケット”のメンバである。クラス”チケット”はクラス”チケットスタブ”のサブクラスであるため、チケット2608は、スーパークラス”チケットスタブ”から定義が引き継がれたプロパティ”way”及び”travelNote”を他のプロパティの中に備えている。チケットスタブ2632はチケット2608から導かれ、チケットスタブ2632のプロパティ”way”及び”travelNote”はそれぞれチケット2608のプロパティ”way”及び”travelNote”に等しい。前述したように、チケット2608（図65）のプロパティ”travelNote”はエージェント150A-2からエージェント150A-1を区別するために使用することができる。同様にエージェント150A-2の現在のスタック（図示しない）は、そのプロパティ”travelNote”がオブジェクト2620であるチケットスタブ（図示しない）を備えている。

【0688】図70はオペレーション”send”がエージェント150Aによって遂行された後のコンピュータネットワーク2500を示している。エージェント150Aはコンピュータプロセスエンジン132Aのデータ部分132-Dに残留している。エージェント150A-1はエンジン132Bのデータ部分132B-D中のプレイス220Bを占めている。同様にエージェント150A-2は、エンジン132Cのデータ部分132C-Dによって実行されるプレイス220Cを占有している。

【0689】オペレーション”send”のアクセスは”プライベート”であり、応答者（レスポnder）のみがオ

ペレーションを要求することをアローワンスする。以下に詳細に説明したアペンディックスAに説明するように、本発明の各々のフィーチャーは、どんな条件のもとにフィーチャーのリクエストがなされうるかを特定するアクセスを備えている。オペレーション”send”のアクセスはプライベートであるから、エージェント150Aのみがエージェント150Aによるオペレーション”send”の遂行を回避することができる。エンジン132Aでさえも、エージェント150Aによるオペレーション”send”の遂行を開始することはできない。

#### 【0690】ディファードクローニング

オペレーション”send”を遂行するのに必要な時間の量及びオペレーション”send”の遂行の間に作り出されるエージェントのクロンによって占有されるスペースの実質的な節減は、”ディファードクローニング”によって実現される。オペレーション”send”の遂行を行なっているエンジンが、同一のエージェントの2以上のクロンが単一の目的エンジンに移送されるべきことを定めた場合に、ディファードクローニングが生ずる。このような場合に単一のクロンが目的エンジンに移送され、余分のクロンは目的エンジンにおいて単一のクロンから導かれる。図71、図74、図75、図76、図77は例示的である。

【0691】エージェント150A（図71）はエンジン132A中のプレイス220Aを占有している。エンジン132Aはコンピュータシステム110A（図示しない）内において行なわれるコンピュータプロセスである。エージェント150Aは、オペレーション”send”を遂行しており、エージェント150Aのクロンのための目的プレイスとしてプレイス220B、220C及び220Dを特定する3つのチケット（図示しない）のリストをオペレーションへのアーギュメントとして供給している。エンジン132Aは通信リンク102AEを横切ってエンジン132Eと通信する。通信リンク102AEはオペレーション”go”に関連して通信リンク102AZ（図34）について説明した通りである。エンジン132Eはエンジン132Aに加えて、中心リンク102FBFを横切ってエンジン132B、132Fと通信している。さらにエンジン132Fは通信リンク102FCDを横切ってエンジン132C、132Dと通信している。プレイス220B、220C及び220Dは、それぞれエンジン132B、132C及び132Dによって解釈されるプロセスである。従ってエージェント150Aのクロンがプレイス220B、220C、220Dに到達するためには、各々のクロンには最初にエンジン132Eに移送されねばならない。

【0692】エンジン132A中においては、単一のクロンすなわちエージェント150Aのエージェント150A-1が作り出される。エージェント150A-1の実行状態は、エージェント150A-1がオペレーシ

ョン” send”を遂行している時に、センドフレーム2902を含む。エージェントの実行状態は以下に及びアベンディックスAに詳細に説明されている。センドフレームはアベンディックスBに詳細に説明されている。センドフレーム2902（図72）は、チケット2904のリストを含み、このリストは、チケット2904B、2904C及び2904Dを含み、これらのチケットは、目的プレイスとして、それぞれプレイス220B、220C及び220Dを特定する。従ってエージェント150Aの単一のクローンであるエージェント150A-1（図71）はそれぞれプレイス220B、220C、220Dに移行するべき別々の3つのエージェントクローンを表している。

【0693】符号化されたエージェント150A-1-E（図73）はアベンディックスBの符号化規則に従ってエージェント150A-1（図71）を符号化した結果である。符号化エージェント150A-1-E（図73）には、目的点、150A-1-E-D1、150A-1-E-D2及び150A-1-E-D3があり、これらは、エージェント150A（図71）の3つのクローンのそれぞれのトランスファー目的点を規定する。エージェント150Aの3つの全てのクローンがエンジン132Eに転送されるので、目的点150A-1-E-D1、150A-1-E-D2、及び150A-1-E-D3（図73）は、エージェント150A（図71）のそれぞれのクローンのトランスファー目的点としてのエンジン132Eを全て規定している。エンジン132Aは、目的点150A-1-E-D1、150A-1-E-D2及び150A-1-E-D3（図73）が各々トランスファー目的点としてのエンジンを規定することと、エージェント150A（図71）の単一のクローンすなわち符号化されたエージェント150A-1-E（図73）をエンジン132E（図74）に移送させることを定める。

【0694】従って、3つの別々のクローンをエンジン132Aからエンジン132Eに転送するのではなく、単一のクローンが、エンジン132Eに転送され、それによってエージェント150Aのクローンによって占有されたエンジン132A中のスペースの量、及びクローンをエンジン132Eに転送するために必要とされる時間の量が比例的に減少される。

【0695】エージェント150A-1は、オペレーション” go”について前述したように、符号化され、エンジン132E（図74）に転送される。エンジン132Eは、エージェント150A-1のセンドフレーム2902のリスト2904を検索する。符号化されたエージェントの構造は、アベンディックスBに示すように標準化されているので、エンジン例えばエンジン132Eは、符号化されたエージェント150A-1をデコードすることなく、リスト2904を検索することができ

る。エンジン132Eは、センドフレーム2902のチケット2904Bに対応する1つのクローンがエンジン132Bに転送されるべきことと、チケット2904C、2904Dに対応する2つのクローンがエンジン132Fを経てそれぞれエンジン132C、132Dに転送されるべきこと、を定める。そのため第2のクローン、エージェント150A-2（図75）が、エンジン132Eによってエージェント150A-1から作り出される。エージェント150A-1が動かされる時に、エージェント150A-1のコピーであるエージェント150A-2も同様に符号化される。エージェント150A-2は、センドフレーム2902-2を含む、センドフレーム2902-2はチケットリスト2904-2を有する。チケットリスト2904-2は、エージェント150A-1のセンドフレーム2902のチケットリスト2904から除去された単一のチケット2904Bを有する。

【0696】エージェント150A-2はエンジン132Bに移送され、エージェント150A-1はエンジン132F（図76）に移送される。チケット2904Bはトリップの目的点としてプレイス220Bを特定し、チケット2904Bはセンドフレーム2902-2のただ1つのチケットであるから、プレイス220Bはエージェント150A-2の目的点であり、エージェント150Aのいかなる他のクローンもエージェント150A-2から作り出されない。エージェント150A-2はエンジン132Bによってデコードされる。エージェント150A-2は、前述したようにプレイス220Bによりオペレーション” entering”の遂行によってプレイス220Bの占有がパーミットされ、エージェント150A-2についてのオペレーション” send”のオペレーションは完了する。

【0697】センドフレーム2902Bは、そのためエージェント150A-2の実行状態から、実行モデルのコンテキストにおいて以下に説明されるように除去される。エージェント150A-1はエンジン132Fに移送される。エージェント150A-1の実行状態の一部分であるセンドフレーム2902には、チケット2902C、2904Dを含むチケットリスト2904が含まれる。従ってエージェント150A-1はそれぞれプレイス220C、220Dに向かって移動しているエージェント150Aの2つのクローンを表している。従って単一のクローンすなわちエージェント150A-1をエンジン132Eからエンジン132Fに移送することによって、前述したように、エンジン132Eとエンジン132Fの間でデータを転送するのに必要な時間とエンジン132Eに格納するスペースとが実質的に節減される。

【0698】前述したように、エンジン132Fにおいては、（i）エージェント150Aの第3のクローンす

なわちエージェント150A-3がエージェント150A-1から形成され、(i) チケット2904Cを含むエージェント150A-3には、センドフレーム(図示しない)が含まれる。エージェント150A-3がそれからコピーされたエージェント150A-1は、符号化され、エージェント150A-3も符号化される。チケット2904Cはエージェント150A-1のセンドフレーム2902から除去される。エージェント150A-3は、エンジン132Cに移送され、エージェント150A-1は、エンジン132D(図77)に移送される。

【0699】エンジン132Cは、チケット2904Cによって規定されるエージェント150A-3のトリップの目的点であるプレイス220Cを含み、エンジン132Dは、チケット2904Dによって規定されるエージェント150A-1のトリップ目的点であるプレイス220Dを含む。前述したように、(i) エンジン132Cは、エージェント150A-3をデコードし、エージェント150A-3には、オペレーション"entering"の実行によってプレイス220Cの占有がパーミットされ、(ii) エンジン132Dはエージェント150A-1をデコードし、エージェント150A-1には、オペレーション"entering"の遂行によってプレイス220Dの占有がパーミットされる。センドフレーム2902(図示しない)は、目的点としてプレイス220Dを特定化する単一のチケットを有しているので、オペレーション"send"はエージェント150A-1について終了する。同様に、エージェント150A-3は、単一のプレイスすなわち、エージェント150A-3によって占有されるプレイスであるプレイス220Cをトリップの目的点として特定するセンドフレームを含むので、エージェント150A-3についてもオペレーション"send"は同様に終了する。改行エージェント150A-1、150A-2及び150A-3の各々は、それぞれのエージェントクロンが行なうトリップを規定する単一のチケットを含むセンドフレームを含む実行状態を有するオペレーション"send"の遂行を完了する。それぞれの単一のチケットは、それぞれのエージェントによるオペレーション"send"の遂行の結果として生じたチケットスタブを導くために前述したように使用される。

【0700】従って、それぞれの目的プレイスにいくつかのクロンを転送する際にクロニングをできるだけ長い間遅らせることによって、オペレーション"send"の遂行において実質的な量のコンピュータの格納スペース及び時間が節減される。

【0701】センドフレーム2902の、エージェント150A-1の転送中の構造を、図30に示す。クラス"センドフレーム"は、プロパティ"チケット"を規定する。センドフレーム2902のプロパティ"チケット"

ト"は、チケット2904B、2904C及び2904Dをアイテムとするリスト2904である。チケット2904B、2904C及び2904Dは、応答エージェント150Aのそれぞれのクロンが行なうそれぞれのトリップを規定する。

【0702】前述したように、エージェント150Aがオペレーション"send"を遂行することによって、一以上のエージェントクロンが作り出され、各々のエージェントクロンは、エージェント150Aの一以上のクロンを表している。各々のエージェントクロン例えばエージェント150A-1(図71)を形成する場合には、そのエージェントクロンにセンドフレーム2902のコピーが含まれる。センドフレームコピーのプロパティ"チケット"は、エージェントクロンによって表されるクロンに対応するチケットのみを包含するように変更される。一例としてエージェント150A-2(図75)は、センドフレーム2902-2を含み、このセンドフレーム2902-2は、チケット2904Bを含み、エージェント150A-1は、センドフレーム2902を含み、センドフレーム2902は、チケット2904C及び2904Dを含む。

【0703】従ってエージェント150A-2は、チケット2904Bによって指定されたプレイスすなわちプレイス220Bに向かって移動しているエージェント150Aの単一のクロンを表している。同様に、エージェント150A-1は、チケット2904C、2904Dによって特定されたプレイスに向かって移動しているエージェント150Aの2つのクロンをそれぞれ表している。

【0704】各々のクロンがそれぞれの目的点に到達したら、センドフレーム2902のプロパティ"チケット"(図30)は、正確に1つのチケットのリストとなる。一例としてエージェント150A-1、150A-2、150A-3(図77)は、フレーム2902、2902-2、2902-3をそれぞれ含み、これらのフレームは各々単一のチケットすなわちそれぞれのチケット2904D、2904B、2904Cを各々含む。従って各々のセンドフレームは、対応するエージェントクロンが行なうトリップを規定する単一のチケットを有している。この単一のチケットは、それぞれのエージェントクロンによるオペレーション"send"の遂行の結果として生成されるチケットスタブを形成するために前述したように使用される。

【0705】従って2以上のエージェントクロンが部分的に同延であるトリップを行なう限り、オペレーション"send"を遂行するエージェントのクロニングを延期(ディファ)することによって、コンピュータの格納スペース及びデータ転送時間が実質的に節減される。オペレーション"go"のコンテキストによって前述したように、あるエージェントによるオペレーション"go"



の遂行の間に例外が投出された場合には、そのエージェントはパーガトリに置くことができる。同様にエージェントクローンによるオペレーション"send"の遂行の間に例外が投出された場合にはそのエージェントクローンはパーガトリに置くことができる。また単一の符号化されたエージェント例えばエージェント150A-1(図74)によって表されるエージェントの複数のクローンの転送が失敗した場合、その場合、符号化されたエージェントによって表されるこの各々のクローンは、前述したように、デコードされ、活性化され、パーガトリに置かれる。

【0706】エージェント間の相互作用： オペレーション"Meet"

同一のプレイスを占有する2つのエージェントは、一方のエージェントが他方のエージェントにあるオペレーションを行ない又はある属性を設定又は質疑することによって、互いに相互作用することができる。リクエストされたオペレーションのアーギュメント及び結果又は設定又は質疑された属性を介して2つのエージェントの間に情報の交換が行なわれる。一例としてエージェント150A、150Bは、エンジン132B(図50)内において実行されているプレイス220Bを占有している。従ってエージェント150A、150Bは、(i) エージェントBがあるフィーチャーを遂行することをエージェント150Aがリクエストするか、または(ii) エージェントAがフィーチャーを遂行することをエージェント150Bがリクエストすることによって、相互作用することができる。

【0707】エージェント150Aは、エージェント150Aがエージェント150Bに対するリファレンス(参照)を持つ場合にのみエージェント150Bがフィーチャーを遂行することをリクエストすることができる。同様にエージェント150Bは、エージェント150Bがエージェント150Aへのリファレンスを含む場合にのみエージェントAがフィーチャーを遂行することをリクエストすることができる。エージェント150Aは、エージェント150A、150Bが占有者であるミーティングプレイスによってオペレーション"meet"を遂行することをリクエストすることによって、エージェント150Bに対するリファレンスを取得する。ここまではプレイス220Bは、エージェント150A、150Bによって占有されるプレイスとしてのみ記述されている。以下の説明の目的のためには、プレイス220Bは、ミーティングプレイス、すなわちクラス"ミーティングプレイス"の1メンバである。ミーティングプレイスについてはアペンディックスAにおいて一層詳細に説明する。

【0708】図78、図79は、オペレーション"meet"のインタフェースを表している。フレーム3100はエージェント150Aの実行状態の一部分である。フ

レームの3100はミーティングプレイス220Bによるオペレーション"meet"の遂行状態を記録する。ミーティングプレイス220Bは、オペレーション"meet"のレスポンスとしてのエージェント150Aの実行状態内において特定される。

【0709】フレームの3100は、現在のスタックであるスタック3102を含む。スタック3102は、オペレーション"meet"(図78)の遂行の直前に、ペティション3160をその上部に備えている。ペティション3160は、オペレーション"meet"の遂行によって消費されたアーギュメントである。

【0710】ペティションすなわちクラス"Petition"のメンバについては以下に、及びアペンディックスAに、一層詳細に説明する。

【0711】フローチャート3200(図80)は、ミーティングプレイス220Bによって遂行されるオペレーション"meet"のインプリメンテーションを示している。エンジン132B(図50)は、ミーティングプレイス220Bによって遂行されるオペレーション"meet"を遂行する際に、ペティション3106(図78)を分析することによって、ペティションの対象であるエージェントを定める。ペティション3106はペティションの対象であるエージェントをネーム、クラス又はその両方によって特定する。ペティションの対象であるエージェントは、ミーティングをリクエストしたエージェントがミーティングを行なうようになっているエージェントとして、ペティションによって特定されたエージェントである。

【0712】エンジン132Bは、ポップペティションステップ3202(図80)において現在のスタックからペティション3106をポップする。処理はポップペティションステップ3202からステップ3204に移行する。ステップ3204においてエンジン132Bは、新しいエンプティなテレネームリストを作り出す。ステップ3204において作り出されたリストのテレネームは、リクエストしたエージェントとのミーティングを拒絶したペティションの対象であるエージェントのテレネームである。最初はペティションの対象であるいかなるエージェントもリクエストしたエージェントとのミーティングを拒絶していないので、このリクエストは最初はエンプティである。

【0713】処理はステップ3204から、ファインドペティションエージェントステップ3206に移行し、このステップ3206において、オペレーション"meet"を遂行しているエンジンがペティションの対象であるエージェントを、すなわち、ステップ3204(図80)によって作り出されたテレネームのリストのアイテムにそのテレネームが含まれない、ペティション3106(図78)を満たすエージェントを見出す。あるエージェントは次のようにしてペティション3106(図

78)を満たす。

【0714】アペンディクスA2に詳細に説明するように、ペティション3106は、プロパティ“agentName”及びプロパティ“agentClass”（どちらも図示しない）を含む。プロパティ“agentName”及び“agentClass”はオプションであり、従ってどちらもゼロであってもよい。ペティション3106のプロパティ“agentName”がテレネームであり、プロパティ“agentClass”がゼロである場合、ペティションの対象であるエージェントは、ペティション3106のプロパティ“agentName”であるテレネームによってそのテレネームが指定されたエージェントである。

【0715】プロパティ“agentClass”がサイティションであり、プロパティ“agentName”がゼロである場合、ペティションの対象であるエージェントは、ペティション3106のプロパティ3106のプロパティ“agentClass”であるサイティションによって特定されたクラスのメンバであるエージェントである。どちらのプロパティもゼロでない場合ペティションの対象であるエージェントは、両方の基準を満たすエージェントである。どちらのプロパティもゼロである場合、クラス“ミーティング不適切”の例外が投出され、オペレーション“meet”を失敗に終わらせる。

【0716】処理がファインドペティションドエージェントステップ3206からテストステップ3208に移行し、このテストステップにおいて、エンジン132Bは、ペティションの対象であるエージェントがファインドペティションドエージェントステップ3206において見出されるか否かを定める。ペティションの対象であるエージェントがファインドペティションドエージェントステップ3206で見出されなかったら、処理がテストステップ3208から、後述するウェイトステップ3222に移行する。逆にペティションに対象であるエージェントがファインドペティションドエージェントステップ3206において見出されたら、処理はテストステップ3208からミーティングステップ3210に移行する。

【0717】ミーティングステップ3210において、ペティションの対象であるエージェントには、ミーティングプレイス220Bを解釈しているエンジン132Bによってオペレーション“meeting”を遂行することが求められる。オペレーション“meeting”を遂行する際に、ペティションの対象であるエージェントは、エージェント150Aとのミーティングに参加することに同意するか、又はこれを拒絶する。オペレーション“meeting”については、以下又はアペンディクスAに詳細に説明されている。

【0718】ペティションの対象であるエージェントによるオペレーション“meeting”の遂行後に処理はミーティングステップ3210からテストステップ3212

に移行する。テストステップ3212においてエンジン132Bは、オペレーション“meeting”が成功のうちに終了し、ペティションの対象であるエージェントがミーティングに同意したことを示すか否かを定める。ペティションの対象であるエージェントはミーティングを拒絶した場合、すなわち処理はテストステップ3212から、以下に説明する第1のフリクオリファイドテストステップ3234に移行する。その逆に、ペティションの対象であるエージェントがミーティングに同意した場合、すなわちオペレーション“meeting”が成功のうちに終了した場合、処理はテストステップ3212からビルドコンタクトステップ3216に移行する。

【0719】ビルドコンタクトステップ3216において、ペティションの対象であるエージェントをサブジェクトするコンタクト3108（図79）が作り出される。処理はビルドコンタクトステップ3216（図80）からプッシュコンタクトステップ3218に移行し、このステップ3218において、コンタクト3108（図79）がスタック3102に保存されることによって、コンタクト3108を生成する。処理はプッシュコンタクトステップ3218から終了ステップ3220に移行しこのステップ3220においてオペレーション“meet”は成功のうちに終了する。

【0720】しかし前述したようにペティションの対象であるエージェントが、オペレーション“meeting”の遂行において反対を表明し、それによってエージェント150Aとのミーティングを拒絶した場合には、処理はテストステップ3212から第1のフリクオリファイドテストステップ3234に移行する。第1のフリクオリファイドテストステップ3234では、エンジン132Bは、ペティション3106（図78）が十分に有資格であるか否かを定める。ペティション3106は、ペティション3106のプロパティ“agentName”であるテレネームが十分に有資格である場合に有資格とされる。テレネームは、プロパティ“authority”とプロパティ“identity”とをともに有する。テレネームについてはアペンディクスAに一層詳細に説明されている。テレネームのプロパティ“identity”はオプションであり、すなわちゼロとすることができる。テレネームのプロパティ“identity”がゼロである場合、そのテレネームは部分的に有資格であり、テレネームのプロパティ“authority”であるオーソリティのすべてのネームのあるオブジェクトを意味する。他方では、テレネームのプロパティ“identity”がゼロでない場合、そのテレネームは十分に有資格であり、正確にゼロ又は1つの面のあるオブジェクトを意味する。

【0721】第1のフリクオリファイドテストステップ3234（図80）において、エンジン132Bは、ペティション3106（図78）が十分に有資格であるか否かを定める。ペティション3106（図78）が十分

に有資格であれば、処理は第1のフリクオリファイドテストステップ3234から終了ステップ3236に移行し、この終了ステップにおいて、クラス“ミーティング拒絶の”反対が表明され、オペレーション“meeting”を失敗に終わらせる。したがって、ペティション3106（図78）が十分に有資格であり、ペティション3106を満たす1つのエージェントがミーティングを拒絶した場合、オペレーション“meet”は失敗に終わる。他方では、ペティション3106を十分に有資格でない場合には、処理は第1のフリクオリファイドテストステップ3234からアッドエージェントツウリストステップ3214に移行する。

【0722】アッドエージェントツウリストステップ3214において、ファインドペティションドエージェントステップ3206において見出された、ペティションの対象であるエージェントが、ステップ3204において作り出されたテレネームリストに付加される。処理はアッドエージェントツウリストステップ3214からファインドペティションドエージェントステップ3206に移行する。

【0723】前述したように、ペティションの対象であるエージェントがファインドペティションドエージェントステップ3206において見出されなかった場合には、処理にはテストステップ3208からウェイトステップ3222に移行する。ウェイトステップ3222においては、エージェント150Aの解釈は、ペティション3106（図78）が終了するか、又はエージェントがミーティングブレイス220Bに入るまで、すなわちミーティング220Bが成功のうちにオペレーション“entering”を遂行するまで中断される。ペティション3106は、アペンディクスAに一層詳細に説明されるように、オペレーション“meet”が成功のうちに、又はそれ以外の仕方でも終了するまで、オペレーション“meet”の遂行の開始から経過した時間の最大量を規定するプロパティ“maximumWait”を有する。ペティション3106は、オペレーション“wait”の遂行の開始からプロパティ“maximumWait”に特定された時間量が経過し終わった時に終了する。

【0724】ペティション3106が、終了するか、又はエージェントがミーティングブレイス220Bに入った時に、エージェント150Aの解釈が再開され、処理がミートステップ3222（図80）からタイムアウトテストステップ3224に移行する。タイムアウトテストステップ3224では、エンジン132B（図50）が、エージェント150A（図78）の解釈の再開がペティション3106の終了の結果であるか否かを定める。エージェント150Aの解釈がペティション3106（図78）の終了の結果として再開された場合には、処理はタイムアウトテストステップ3224（図80）から終了ステップ3226に移行し、このステップ32

26において、クラス“Petition Expired”の例外が表明され、オペレーション“meet”を失敗に終わらせる。

【0725】他方では、エージェント150A（図78）の解釈が、エージェントがブレイス220Bに入ったことの結果として再開された場合には、処理はタイムアウトステップ3224からテストステップ3228に移行する。テストステップ3228では、エンジン132B（図50）は、ブレイス220Bに入ったエージェント（図示しない）がペティションの対象であるか否か、すなわちペティション3106（図78）を満たし、ステップ3204（図80）において作り出されたテレネームのリストにないか否かを定める。エンタしたエージェントがペティション3106（図78）を満たさないか、又はエンタしたエージェントのテレネームがステップ3204（図80）において作り出されたテレネームのリストのアイテムである場合には、処理はテストステップ3228からウェイトステップ3222に移行する。

【0726】他方では、エンタしたエージェントがペティション3106（図78）を満たし、エンタしたエージェントのテレネームが、ステップ3204（図80）で作り出されたテレネームのリストのアイテムでない場合、処理はテストステップ3228から第2のミーティングステップ3230に移行する。第2のミーティングステップ3230では、エンタしたエージェントは、オペレーション“meeting”を遂行するように指示される。処理は第2のミーティングステップ3230からテストステップ3232に移行し、このテストステップ3232においてエンジン132B（図50）は、エンタしたエージェントによるオペレーション“meet”の遂行が成功し、エンタしたエージェントがそれによってエージェント150Aとミートを受け入れたか、又はオペレーション“meeting”の遂行が失敗し、エンタしたエージェントがそれによってエージェント150Aとミーティングを拒絶したか否かを定める。

【0727】エンタしたパーティによるオペレーション“meeting”の遂行が成功した場合、処理はテストステップ3232からビルドコンタクトステップ3216に移行する。前述したように、ビルドコンタクトステップ3216及び後続するステップにおいて、ペティションの対象であるエージェントすなわちエンタしたエージェントとコンタクトが作り出され、そのコンタクトが現在のスタックに保存され、オペレーション“meeting”は成功のうちに終了する。他方では、エンタしたエージェントによるオペレーション“meeting”の遂行に対して例外が表明された場合には処理は、テストステップ3232から第2のフリクオリファイドテストステップ3238に移行する。

【0728】第2のフリクオリファイドテストステップ3238において、エンジン132B（図50）は、ペ

ティション3106（図78）が十分に有資格であるか否かを定める。ペティション3106が十分有資格である場合、処理は第2のフリクオリファイドテストステップ3238（図80）から終了ステップ3240に移行し、この終了ステップにおいて、クラス“ミーティング拒絶”の例外が表明され、ミーティングプレイス220Bによって行われるオペレーション“meeting”を失敗に終わらせる。

【0729】その逆に、ペティション3106（図78）が十分に有資格でない場合、処理は第2のフリクオリファイドテストステップ3238から第2のアドエージェントツウリストステップ3242に移行する。第2のアドエージェントツウリストステップ3242では、ペティションの対象であるエージェントすなわちエンタしたエージェントのテレネームが、ステップ3204において作り出されたテレネームのリストに付加される。処理は第2のアドエージェントツウリストステップ3242から、前述したステップ3222に移行する。

【0730】このようにしてエージェント150Aとペティションの対象であるエージェントとの間のミーティングが取り決められる。このペティションの対象であるエージェントは、（i）ペティション3106において特定されたネーム及びクラスを用いて（ii）応答するミーティングプレイスを占有し、（iii）ペティション3106において特定された最大の時間内においてミーティングを行うことに同意したエージェントである。この最後に述べた条件は、第1のエージェントによって占有されるミーティングプレイスを占有していないが、ある時間内にそのミーティングに到着することが期待されている第2のエージェントとのミーティングを第1のエージェントがリクエストすることを許容する。

【0731】オペレーション“meeting”（図79）の遂行の直後に、スタック3102は、その上部に、前述したコンタクト3108を有している。コンタクト3108は、エージェント150B（図50）へのリファレンスであるプロパティ“subject”を備えている。コンタクト3108（図79）のプロパティ“subject”はコンタクト3108のアトリビュート“subject”を指示することによって生成される。コンタクトのアトリビュート“subject”についてはアペンディクスAに詳細に説明されている。したがって、エージェント150Aは、エージェント150B（図50）への参照を取得し、したがってエージェント150Bがフィーチャを遂行することをリクエストすることができる。

【0732】オペレーション“meeting”のコンテキストにおいて以下に説明するように、エージェント150Bは、ミーティングをリクエストするエージェントとしてエージェント150Aを特定化するコンタクトを、オペレーション“meeting”の遂行においてのアーギュメ

ントとして消費する。ミーティングプレイス220Bによるオペレーション“meeting”の遂行の後に、エージェント150Bによって消費されたコンタクトは、エージェント150Aに対するリファレンス（参照）であるプロパティ“subject”も備えている。したがってエージェント150Bは、エージェント150Aのリファレンスを取得し、エージェント150Aがフィーチャを遂行することをリクエストすることができる。

【0733】コンタクトされるエージェントは、ミックスインクラス“コンタクテッド”が受け継がれたクラスのメンバである。以下にアペンディクスAに詳細に説明するように、ミックスインクラス“コンタクテッド”は、プロパティ“contacts”へのアクセスを提供するアトリビュート“contacts”を規定する。プロパティ“contacts”は、コンタクト（複数）のセットである。エージェント150B（図81）は、コンタクトされたエージェントであり、したがってセット3320であるプロパティ“contacts”を備えている。前述したようにミーティングプレイス220Bによるオペレーション“meeting”が遂行された後は、エージェント150Bによるオペレーション“meeting”の遂行において消費されたコンタクトであるコンタクト3404は、エンジン132Bによってセット3302に付加される。

【0734】ペティションの対象であるエージェントの同意：オペレーション“Meeting”

前述したように、ペティションの対象であるエージェントによるオペレーション“meeting”遂行によるミーティングをリクエストしたエージェントとミーティングに同意したり拒絶したりする。オペレーション“meeting”にはクラス“Agent”によって規定されても受け継がれてもいない。その代わりに、オペレーション“meeting”は、ミックスインクラス“Petitioned”によって規定されている。クラス“Agent”は、ミックスインクラス“Petitioned”のサブクラスではなく、本発明のユーザーによって後に規定されるクラス“Agent”のサブクラスは、ミックスインクラス“Petitioned”のサブクラスであることができる。したがって、このようなユーザーによって規定されるサブクラスのメンバであるエージェントのみが、オペレーション“meeting”を遂行することができ、したがって他のエージェントとのミーティングに参加することができる。

【0735】図82は、エージェント150Bによるオペレーション“meeting”の遂行の直前におけるエージェント150Aの実行状態を示している。クレーム3400は、エージェント150Aの実行状態の一部分であり、エージェント150Bが実行するオペレーション“meeting”の動的状態を記録する。オペレーション“meeting”が、エンジン132Bによってリクエストされる。オペレーション“meeting”の遂行は、エージェント150Aの実行状態において記憶される。エー

ジェント150A又は好ましくはエージェント150Bのパーミットは、オペレーション”meeting”の遂行によって喪失され得る。エージェント150B（図50）は、オペレーション”meeting”のレスポンドとして、エンジン132B中において特定化される。

【0736】スタック3402は、フレーム3400のプロパティ”stack”であり、現在のスタックである。オペレーション”meeting”の遂行の直前に、スタック3402は、上部から下部にかけて、コンタクト3404及びペティション3406を有している。

【0737】コンタクト3404は、ミーティングをリクエストするエージェントとしてエージェント150Aを特定化する。コンタクト3404のプロパティ”subjectName”（図示しない）は、エージェント150Aのテレネームと同等のテレネームであり、それによってエージェント150Aをリクエストするエージェントとして特定化する。コンタクト3404のプロパティ”subjectClass”（図示しない）はエージェント150Aが1つのインスタンスであるクラスを特定化するサイティションである。通常エージェント150Aのリファレンスであるコンタクト3404のプロパティ”subject”

（図示しない）は、エンジン132Bによって0にされる。したがってエージェント150Bは、オペレーション”meeting”の遂行においてコンタクト3404を消費することによって、ミーティングをリクエストするエージェントとしてエージェント150Aを適切に特定化する上に必要なすべての情報をもつが、エージェント150Aへのリファレンスは持たず、したがってエージェント150Aと相互作用する方策を持たない。このように、どちらのエージェントも、他のエージェントと、両者がミーティングに同意するまでは、相互作用することができない。

【0738】ペティション3406は、ミーティングをリクエストする際にエージェント150A（図50）によって供給されるペティション3106（図78）のコピーである。それはこのアーギュメントが”バイコピー”をパスするからである。それはこのアーギュメントがパスした”バイコピー”だからである。アーギュメント”バイコピー”の通過については以下にまたアペンデックスAに詳細に説明する。

【0739】エージェント150Bは、オペレーション”meeting”を成功のうちに遂行することによって、コンタクト3404によって特定化されたエージェントとペティション3406（図82）によって規定されたミーティングを行うことに同意する。このミーティングは、例外を表明することにより、拒絶され、それによってオペレーション”meeting”は失敗に終わる。ミックスインクラス”Petitioned”によって否定されたオペレーション”meeting”の方法は決して成功せず、常にクラス”ミーティング拒絶”の例外を常に表明する。しか

し、ミックスインクラス”Petitioned”から受け継がれたクラス”Agent”の後に規定されたサブクラスは、ある事情の元においては成功するように、オペレーション”meeting”のインプレメンテーションを再定義することができる。

【0740】エージェントは広い範囲のニーズに応え、広い範囲のサービスを遂行するようにするものであることが予想されるので、クラス”Agent”のサブクラス及びオペレーション”meeting”の方法の変形の数是非常に大きい。一例として、論理フローダイアグラム3500（図83）はオペレーション”meeting”のための1つのそのような方法を示している。ステップ3502において、コンタクト3404（図82）及びペティション3406はスタック3402からポップ（復元）される。処理はステップ3502（図83）からステップ3504に移行し、このステップ3504において、コンタクト3404（図82）のアトリビュート”subjectClass”が質疑されることによって、コンタクト3404のプロパティ”subjectClass”が生成する。処理はステップ3504（図82）及びペティション3406はスタック3402からポップされる。処理はステップ3502（図83）からステップ3504に移行しこのステップ3504においてコンタクト3404（図82）のアトリビュート”subjectClass”が質疑されることによってコンタクト3404のプロパティ”subjectClass”が発生する。処理はステップ3504（図83）からテストステップ3506に移行し、このテストステップにおいてコンタクト3404（図82）のプロパティ”subjectClass”が、特定のクラスのサイティションと比較される。コンタクト3404のプロパティ”subjectClass”がサイティションに等しい場合、処理はテストステップ3506（図83）から終了ステップ3508に移行する。終了ステップ3508においてオペレーション”meeting”は成功のうちに終了し、それによってミーティングに対する同意が得られる。さもないときは、コンタクト3404（図82）のプロパティ”subjectClass”がサイティションに等しくない場合、処理はテストステップ3506（図83）から終了ステップ3510に移行し、この終了ステップ3510においてクラス”ミーティング拒絶”の例外が表明されることによって、オペレーション”meeting”が失敗に終わりそれによって、コンタクト3404（図82）によって特定化されたエージェントとのミーティングが拒絶される。

【0741】前述したように、オペレーション”meeting”の実行のリクエストにおいてアーギュメントとしてエージェント150B（図50）に供給された、エージェント150Aを特定化するコンタクト3404は、エージェント150Aへのリファレンスを含まない。その代わりに、コンタクト3404（図82）のプロパティ”subject”は0である。前述したように、ミーティ

ングプレイス220B(図50)によるオペレーション”meeting”の遂行によって、ミーティングが成功のうちに取引終われると、エンジン132Bは、コンタクト3404(図82)のプロパティ”subject”をエージェント150Aへのリファレンスとなるように変更する。

【0742】このように、どんな事情のもとでどのエージェントとともに、特定のエージェントがミーティングに同意するように構成されているかを定めるために、実質的な量のデータを処理する複雑なロジックを適用するために、アペンディクスAに述べたコンピュータインストラクションセットの十分な一般性及びデータ処理能力を使用することができる。

【0743】図84は、エージェント150B(図50)によるオペレーション”meeting”の遂行の直後のフレーム3400の状態を示している。スタック3402(図84)は、オペレーション”meeting”がいかなる欠陥をもたらさなかったのかでエンブティである。オペレーション”meeting”のアクセスは”システム”であり、ある1つのエンジンのみがオペレーション”meeting”をリクエストすることができる。なお、ミックスインクラス”Petitioned”に規定されたオペレーション”meeting”は、以上に説明したアペンディクスAに説明されるクラス”ミーティングプレイス”に規定されたオペレーション”meeting”とは別個のものである。

【0744】エージェント間の相互作用の停止：オペレーション”Part”

前述したように、2つのエージェントは、それらの間のミーティングの間に、どちらかの間に、相互及びどちらかのエージェントによって所有されるオブジェクトへのリファレンスを交換する。2つのエージェントの間の相互作用を停止するにはどちらのエージェントも他のエージェントへのリファレンスも他のエージェントが所有するオブジェクトへのリファレンスも持たないことを確実にすることが必要とされる。

【0745】あるミーティングプレイスを占有する2つのエージェントの間のミーティングに参加するどのエージェントも、ミーティングプレイスによるオペレーション”part”の遂行の要求によってそのミーティングを終了させることができる。

【0746】ミーティングプレイス220Bによるオペレーション”meeting”の成功した遂行によってエージェント150Aは、エージェント150Bへのリファレンス150A-R1を取得しており、またエージェント150Bは、エージェント150Aへのリファレンス150B-R1を取得している。オブジェクト3602はエージェント150Aによって所有されている。同様にオブジェクト3604は、エージェント150Bによって所有されている。図85は、エージェント150A、150Bが、相互作用しオブジェクト3602、360

4へのリファレンスを交換した後の、エージェント150A、150Bの状態を示している。エージェント150Aは、オブジェクト3602へのリファレンスをエージェント150Bに与えることによって、オブジェクト3602へのアクセスをエージェント150Bに許容する。リファレンス150B-R1、150B-R2によるこのようなアクセスによって、エージェント150Bは、エンジン132Bに命令を送出することができ、それによってエージェント150A、オブジェクト3602は、送出された命令に従って処置をそれぞれとることができるようになる。同様にエージェント150Bは、オブジェクト3604へのリファレンス150A-R2をエージェント150Aに与えることによって、オブジェクト3604への同様なアクセスをエージェント150Aに許可する。

【0747】エージェント150Aは、それぞれエージェント150B、オブジェクト3604、オブジェクト3602へのリファレンス150A-R1、150A-R2及び150A-R3を有している。リファレンス150A-R1、150A-R2及び150A-R3は、

(i) スタック中に含まれているか、又は、エージェント150Aの実行状態の一部分であるフレーム(図示しない)の変数リスト(やはり図示しない)内に収容されているか、又は(i) エージェント150Aのプロパティ又はプロパティの成分として格納されている。エージェントの実行状態については以下に説明する。エージェント150Bは、同様に、エージェント150A、オブジェクト3602及びオブジェクト3604へのそれぞれのリファレンス150B-R1、150B-R2及び150B-R3を有している。エージェント150Bは、オブジェクト3604へのリファレンス150A-R2をエージェント150Aに与えることによって、エージェント150Bは、(i) オブジェクト3604にあるオペレーションを遂行することをリクエストしたり、(i) オブジェクト3604の一部又は全部をコピーしたりすることをエージェント150Aに許容する。

【0748】エージェント150A又は150Bは、ミーティングプレイス220Bがオペレーション”part”を行うことをリクエストすることによって、エージェント150A、150Bの間の相互作用を終了させることができる。説明の目的のために、エージェント150Aは、ミーティングプレイス220Bがオペレーション”part”を行うことを求めるエージェントである。図86、図87は、それぞれミーティングプレイス220Bによるオペレーション”part”の遂行の直前及び直後のエージェント150Aの状態を示している。フレーム3702は、エージェント150Aの実行状態の一部分であり、ミーティングプレイス220Bによって遂行されるオペレーション”part”の状態を記録する。ミーティングプレイス220Bは、フレーム3702のプロパティ

ィ” responder”であり、したがってオペレーション” part”のレスポンスである。スタック3704は、エージェント150Aの実行状態の一部分でもあるフレーム3702のプロパティ” stack”であり、現在のスタックである。

【0749】スタック3704の上部にはコンタクト3706がある。コンタクト3706は、エージェント150Aがそれから分離されるべきエージェントとしてエージェント150Bを特定化している。エンジン132B（図50）は、ミーティングプロパティ220Bのためにオペレーション” part”を実行している間に、オペレーション” part”を行うようにエージェント150Bに指示する。エージェント150Bはエージェント150Bが分離しようとしているエージェントとしてエージェント150Aを特定化するコンタクトを単一のアーギュメントとして消費する。通常エージェント150Aへのリファレンスであるコンタクトのプロパティ” subject”は、オペレーション” part”をリクエストする前にエンジン132Bによってボイドされるので、エージェント150Bは、ミーティングの終了後にエージェント150Aへのリファレンスとともに残されることはない。オペレーション” part”を行う際に、エージェント150Bはいかなる結果も生成させない。オペレーション” part”の遂行の動的状態は、エージェント150B（図50）又はエージェント150Aの実行状態の一部分ではなく、その代わりに、エンジンプロセスの実行状態の一部分である。オペレーション” part”の実行によって表明された例外は、エージェント150B又はエージェント150Aによって経験されず、したがってこれらのエージェントに対していかなる経過もいかなる効果も持たない。エージェント150Bは、オペレーション” part”を行う際に、エージェント150A、150Bの間のミーティングがエージェント150Aによって終了されたことの通知を受ける。エージェント150Bがコンタクトされるエージェントである場合、エージェント150Aを参照するコンタクトであるエージェント150Aのプロパティ” contacts”のアイテムは、エージェント150Bのプロパティ” contacts”からリクエストされる。オペレーション” part”のアクセスは”システム”であるため、エージェント150Bがオペレーション” part”を行うことをリクエストし得るのはエンジン132Bのみである。

【0750】図87は、ミーティングブレイス222Bによるオペレーション” part”の遂行の直後においてのエージェント150Aの状態を示している。オペレーション” part”がいかなる結果ももたらさないのので、スタック3704はエンptyである。

【0751】エージェント150A、150Bの間のミーティングの終了は、エージェント150B及びエージェント150Bによって所有されるすべてのオブジェク

トへのエージェント150A中のすべてのリファレンスをボイドにする。一例として、図88において、エージェント150Bを既に特定化したリファレンス150A-R1とオブジェクト3604を既に特定化したリファレンス150A-R2はどちらもボイドされる。同様に、エージェント150Aを以前にリファレンスしたリファレンス150B-R1及びオブジェクト3604を既にリファレンスしたリファレンス150B-R2も、エージェント150B中においてボイドされる。したがって、エージェント150A、150Bは、もはや両方を交換することはできない。エージェント150A、150Bが、コンタクトされるエージェントである場合、エージェント150Aを特定化するコンタクトは、エージェント150Bのプロパティ” contacts”から除去され、エージェント150Bを特定化するコンタクトは、エージェント150Aのプロパティ” contacts”から除去される。

#### 【0752】大規模ワイドエリアネットワークへの本発明の応用可能性

これまでの、オペレーション” go”又はオペレーション” send”の実行によって、エージェント150A（図34）がミーティングブレイス220Bにそれ自身を転送したり又はエージェント150A自身のクローンがミーティングブレイス220Bに転送されたりするコミュニケーションシステムが記述された。この場合、エージェント150Aは、エージェント150A、150Bとの間のミーティングを取り決めるためにミーティングブレイス220Bによってオペレーション” meeting”を遂行することをリクエストする。このミーティングの間にエージェント150A、150Bは、以下に一層詳細に説明するように情報を交換することができる。

【0753】以上の説明は、本発明の非常に簡単な使用に関するものであるが、本発明についてはより一般的な使用も可能である。本明細書中及びアペンディクスAに記述されるコンピュータ命令のセットの適応によって、あるエージェントは、他のブレイスにそれ自身を転送させまたその他のエージェント及びブレイスとミーティングして情報を交換するべきかを定めるために、非常に複雑で込み入ったロジックを適用することができる。また本発明は、以上に説明したように、2又は3のコンピュータシステムのネットワークが限定される、また同質的なネットワークにも限定されない。本明細書中及びアペンディクスAに記述されるコンピュータ命令のセットは、広いエリアのネットワークを通るエージェントを作り出すことができるように広い面積のネットワークにわたってブレイスを作り出しそして位置決めするために使用することができる。ここに管理された命令セットは、実質的なネットワーク中にインプリメントすることができるので、広い面積のネットワークは、大型のメインフレームコンピューター、ローカルエリアネットワーク及



びその広い面積のネットワークにおいてエージェントを移動させることができる小型のパーソナルコンピュータを含む広い範囲のコンピュータシステムを組むことができる。さらに2つのエージェントの間のエンタアクションとしてミーティングを定義したが、エージェントはどんな数のミーティングにも同時に参加することができるので、多数のエージェントが最初に述べたエージェントが占有するものと同一のミーティングプレイスを占有する限り、これらの多数のエージェントと同時に相互作用することができる。

【0754】エージェント間の相互作用－実行モデルへのイントロダクション

エージェント150A（図50）は、エージェント150Aとエージェント150Bとの間のミーティングの間に、あるフィーチャを遂行するようにエージェント150Bに指示する命令を送出することによって、エージェント150Bと相互作用する。あるフィーチャを遂行がリクエストされるメカニズムについて以下に説明する。このメカニズムについては以下にそしてアペンディクスAにおいて一層詳細に説明される。

【0755】実行状態は、各々のプロセスに関係している。用語集において前述したように、各々の方法は、その方法の実行の間ある動的状態を有する。あるプロセスの実行状態は、1以上のフレームを含み、各々のフレームは、そのプロセスの実行状態の一部である方法の動的状態を記録する。各々のフレームにはスタックがありこのスタックには、オペレーションのリクエストの前にアーギュメントを保存することができたこのスタックから、オペレーションの実行後に結果を復元することができる。現在のフレームのスタックは現在のスタックである。現在のフレームは、エンジンによって実行されているコンピュータ命令を含むフレームである。

【0756】あるオペレーションをリクエストする前にリクエストは、0又は1以上へのリファレンスを現在のスタックに保存し、これらのオブジェクトは、それによって、オペレーションのためのアーギュメントとして供給される。リクエストは次にレスポンドへのリファレンスを現在のスタックに保存させる。レスポンドは、リクエストされたオペレーションを実行するようにリクエストによって指示されたオブジェクトである。最後に、リクエストは、リクエストされたオペレーションを特定する識別子の実行をリクエストする。エンジンは、リクエストするオペレーションを実行する際に、現在のスタックの上部からオブジェクトをポップ（復元）し、そのオブジェクトをオペレーションのレスポンドとなる。そのオペレーションをインプリメントする方法は、レスポンドがメンバであるクラス内に見出され、そして実行される。一実施例によれば、方法の遂行は、（i）新しいスタックを形成し、（i i）現在のスタックからオペレーションのアーギュメントを復元してそれらを新しいスタ

ックに保存させ、（i i i）新しいスタックを現在のスタックとし、（i v）新しい現在のスタックからアーギュメントを復元し、（v）そのインプリメントによって結果が得られた場合、その結果を新しい現在のスタックに保存させ、そして（v i）その結果（もしあれば）を新しい現在のスタックから復元し、結果（もしあれば）を以前に現在のスタックであったスタックに保存させる。全体としてのオペレーション及びフィーチャの実行については以下にそしてアペンディクスAにおいて一層詳細に説明されている。

【0757】以下の例はオペレーションがリクエストされるメカニズムを表している。前述したように、ミーティングの間に、エージェント150Aは、オペレーションを遂行するようにエージェント150Bに指示することによってエージェント150Bと相互作用する。エージェント150Aは、リクエストされたオペレーションへのアーギュメントとしてエージェント150A中に含まれるオブジェクトを供給することによって、これらのオブジェクトにアクセスすることをエージェント150Bに許容する。エージェント150Bは、エージェント150Bに含まれるあるオブジェクトを、リクエストされたオペレーションの結果として生成させることによって、このオブジェクトへのアクセスをエージェント150Aに許容する。図89-39Fは、エージェント150Bと相互作用して情報をこのエージェント150Bに送出するエージェント150A（図50、図89～図94には示さない）の例を示している。

【0758】図89はエンptyなスタック3902を示しているスタック3902は、エージェント150A（図示しない）の実行状態の現在のスタックである。図89はスタック3902をエンptyであるものとして示しているが必ずしもそうでなくてもよい。しかし図89のスタック3902のいかなるオブジェクトもこの例によっては影響されない。”これはメッセージである”というテキストを有する文字列3904はスタック3902（図90）に保存される。エージェント150Bへのリファレンスは、スタック3902（図91）に保存される。

【0759】エージェント150Aは、次に、”store”をテキストとし、オペレーション”store”にリファレンスする識別子を次に実行する。識別子の実行に際して、あるオブジェクトすなわちエージェント150Bは、スタック3902から復元され、そのオブジェクトは、オペレーション”store”（図92）を実行するように指示される。この例において、オペレーション”store”はあるクラスについて定義される。エージェント150Bはこのクラスのメンバであり、このクラスの方法は、フローチャート4000（図95）によって示される。

【0760】オペレーション”store”を遂行する際

に、エージェント150Bは、ステップ4002（図95）においてスタック3902から文字列3904を復元する。エージェント150Bは、オペレーション”store”を遂行するものとして記述されているが、オペレーション”store”のエージェント150Bによる実行の動的状態を記録するフレーム（図示しない）は、エージェント150A（図示しない）の実行状態の一部である。チャージズアラウワンスすなわちエージェント150Aのパーミットのプロパティ”charges”はそれによって、エージェント150Bによるオペレーション”store”の遂行から結果する処理のために送出される。エージェント150Bは、オペレーション”store”を遂行するオブジェクトとなるが、これは、（i）エージェント150A（図示しない）の実行状態がエージェント150Bをレスポンドとして特定化し、（ii）オペレーション”store”の方法が、エージェント150Bがそのメンバであるクラスによって提供され、（iii）エージェント150Bの内部状態が、オペレーション”store”の遂行されているコンテキストを提供するからである。

【0761】処理はステップ4002からステップ4004に移行し、このステップ4004において、文字列3904のコピーである文字列3904-Cが形成され、図93に示した状態となる。文字列3904、3904-Cは、エージェント150Aの実行状態の一部であり、エージェント150Bによって遂行されたオペレーション”store”の動的状態を記録している、フレーム3906内に含まれている。処理はステップ4004からステップ4006（図95）に移行し、ここでエージェント150Bは、文字列3904-Cを格納する。処理はステップ4006からステップ4008に移行し、ここでエージェント150Bは、文字列3904を廃棄し、この文字列はスタック3902からポップされる。処理はステップ4008から終了ステップ4010に移行し、ここでオペレーション”store”は成功のうちに終了する。ステップ4008によって図94に示した状態となる。すなわちエージェント150Aは、情報をエージェント150Bに成功のうちに移送させている。

【0762】このようにエージェント150Aは、ある操作を遂行するようにエージェント150Bにリクエストすることによって、エージェント150Bと相互作用し、エージェント150Aは、リクエストされたオペレーションへのアーギュメント又は実行されたオペレーションの結果としてオブジェクトをエージェント150Bに送出する。エージェント150Bがエージェント150Aのリクエストに従ってオペレーションを遂行するように以上に説明したが、あるオブジェクトによるあるフィーチャの実行は、実際には、そのオブジェクトの内部状態及びクラスによって規定されたそのオブジェクトの

コンテキスト内においてのエンジンによるフィーチャの遂行である。特別なフィーチャの挙動は、そのフィーチャを実行しているオブジェクトの内部状態に依存して変化することができる。ここにオブジェクトの内部状態は、オブジェクトのプロパティによって部分的に定義される。

【0763】エージェント150Bは、たとえばステップ4008のあと、終了ステップ4010の前に、スタック3902にオブジェクトを保存することによって、オブジェクトをエージェント150Aに移送させることもできる。そのオブジェクトはエージェント150Aによるオペレーション”store”の結果としてスタック4010からポップされる。さらに、オペレーション”meet”について前述対応に、エージェント150Bは、エージェント150Aのリファレンスを含むため、エージェント150Bは、あるオペレーションを行うようにエージェント150Aに指示するとともに、そのオペレーションへのアーギュメントととしてエージェント150Aにオブジェクトを供給することもできる。

【0764】本発明のコンピュータ命令セットの可搬性  
本発明の多くの局面は、非常に多くの可動度及び融通性をプロセスに許容する。第1に、本発明のコンピュータ命令セットは、均質的又は異質的なネットワークにおいて均質的にインプリメントされる。第2に、クラスは、コンピュータ命令のオブジェクトとであるので、可動のプロセスとともにネットワークを通して移動することができる。第3に、コンピュータ命令セットは解釈される。

【0765】コンピュータ命令セットの同質性  
前述したように、データ及びコンピュータ命令を含むエージェントは、1つのコンピュータシステムから次のコンピュータシステムに移動することができ、そしてエージェントがその中に含まれるどんなコンピュータシステムに対しても実行可能である。さらに、エージェントがその内部において移動することのできるネットワークは、均質でも異質でもよい。エージェントが可動であるため、エージェントのプロシージャ部分を形成する各々のコンピュータ命令をネットワークのどのコンピュータシステムが実行しているかを正確に定めることは必ずしもできない。従ってエージェントがその内部を通して移動するネットワークの各々のコンピュータシステムが同一のコンピュータ命令セットを指示していることが大切である。

【0766】開示されたコンピュータ命令セットにおいてオブジェクトとして表されるクラス

第1のコンピュータシステムにおいて実行されているエージェントは、以前に存在していなかった新しいオブジェクトクラスを作りだすことができる。エージェントは、その場合第2のコンピュータシステムにおいて実行中のエンジンに向かって移動することができ、新しく形成さ

れたクラスのメンバの処理を含むエージェントの解釈は、その第2のコンピュータシステム内において続けられる。この場合、第2のコンピュータシステムにおいて実行されているエンジンは、新しく形成されたクラス又はそのクラスのいかなるメンバとも以前に遭遇したことはない。したがって、クラスは、開示された命令セットのオブジェクトとされるので、クラスは、1つのコンピュータシステムから別のコンピュータシステムにエージェントとともに移動することができる。

#### 【0767】クラス構造

いくつかのオブジェクトは、組み合わされて、クラスすなわちクラスオブジェクト、識別子及びクラス定義を限定したり表現したりする。クラスオブジェクト、識別子及びクラス定義は、クラス“class”、“Identifire”及び“Class Definition”のメンバである。

【0768】クラスオブジェクトすなわちクラス“class”のメンバは、そのフィーチャが同一のインターフェイス及びインプリメンテーションを持つオブジェクトセットを表すオブジェクトである。オブジェクトセットはそのクラスの“インスタンス”である。クラス“class”及びクラスオブジェクトについては以下にそしてアペンディクスAに一層詳細に説明される。

【0769】識別子すなわちクラス“Identifire”のメンバは、あるクラスをリファレンスするために、インターフェイスオブジェクト又はインプリメンテーションオブジェクト内において使用される。なお、識別子は、他の形式のオブジェクトにもリファレンスする。インターフェイスオブジェクト及びインプリメンテーションオブジェクトについては以下に一層詳細に説明する。第1のインターフェイスオブジェクト又はインプリメンテーションオブジェクト内の第1のクラスにリファレンスする第1の識別子は、第1のインターフェイスオブジェクト又はインプリメンテーションオブジェクト内の他のすべてのクラスにリファレンスする識別子とは異なっている。しかし、第2のインターフェイスオブジェクト又はインプリメンテーションオブジェクトは、第2のクラスにリファレンスするために、第1の識別子と同等の識別子を使用することができる。

【0770】インターフェイスオブジェクトとインプリメンテーションオブジェクトとは一緒に、フィーチャすなわちオペレーション及び属性のナンバを規定する。オブジェクトの内部状態の一部を提供し、又は変更するためにあるオブジェクトが差し向けられる属性とある数のコンピュータ命令を実行するためにあるオブジェクトが差し向けられるオペレーションとの間の区別は、第1の義的に、本発明の概念形成を助ける上に重要である。属性とは、概念的には、あるオペレーションの特別のケースである。一例としてあるオブジェクトの属性に質疑すると、そのオブジェクトは、そのオブジェクトの内部状態についての情報をそのオブジェクトに供給させるコン

ピュータ命令を実行するように指示される。

【0771】あるオブジェクトによるあるフィーチャの遂行が実際には、そのオブジェクトの内部状態及びクラスによって規定されたそのオブジェクトのコンテキストにおいてのエンジンの特徴の遂行であることを想起されたい。ある特別なフィーチャの挙動は、そのフィーチャを遂行するオブジェクトの内部状態に依存して変化することができる。ここにオブジェクトの内部状態は、部分的に、オブジェクトのプロパティによって規定される。

【0772】単一の識別子は、別々のクラスによって各々規定される2つの別々のフィーチャを同時に特定化することができる。一例としてオペレーション“add”はクラス“ディクショナリ”によるとは異なった仕方です。クラス“ナンバ”によって定義される。あるフィーチャを遂行しているエンジンは、インターフェイス及びインプリメンテーションを選択する際に、そのフィーチャを遂行するオブジェクトのクラスにとって適切なフィーチャの定義及びそれに関連した方法オブジェクトを選定する。換言すれば、フィーチャ定義は、レスポンドをメンバとするクラスから検索される。ある特徴を規定する具体的なクラスは、そのフィーチャのインターフェイスとそのフィーチャのインプリメンテーションとの両方を規定する。あるフィーチャを規定する抽象的なクラスは、そのフィーチャのインターフェイスを規定し、またそのフィーチャのインプリメンテーションを定義することができる。

【0773】あるフィーチャの“インターフェイス”は、そのフィーチャの遂行において消費されたアーギュメント及び生成した結果（もしあれば）を規定する。さらに、あるフィーチャによって生成される例外は、そのフィーチャのインターフェイスによって規定される。あるクラスによって規定されたフィーチャのインターフェイスは、インターフェイスオブジェクトすなわちクラス定義オブジェクトのプロパティ“interface”である。クラス“interface”のメンバによって集散的に表され規定される。クラス“interface”については以下に一層詳細に説明する。

【0774】あるフィーチャのインプリメンテーションはそのフィーチャの“方法”によって限定される。あるフィーチャの方法は、取られるべき処置、すなわちそのフィーチャの実行において実行されるべきコンピュータ命令を規定する。方法オブジェクト、クラス“方法”のメンバは、一般に、一連の命令であり、これらの命令の実行は、そのフィーチャの実行を構成する。あるクラスによって規定されるか、又は受け継がれたフィーチャのインプリメンテーションは、インプリメンテーションオブジェクトすなわちクラス定義オブジェクトのプロパティ“implementation”である。クラス“implementation”のメンバによって集散的に表され規定される。クラス“implementation”については以下に一層詳細に説明す

る。

【0775】フィーチャの定義は、クラスのスーパークラスから受け継がれる。換言すれば、あるフィーチャがある特別なクラスによって定義される場合、そのフィーチャはそのクラスのサブクラスについても定義される。しかしサブクラスはフィーチャのインプリメンテーションを再定義することができ、それによってスーパークラスから受け継がれたインプリメンテーションに代替される。しかしフィーチャはそれについて定義されたサブクラスのスーパークラスによってそのフィーチャが”シール”されていたら、サブクラスは、スーパークラスから受け継がれたインプリメンテーションを再定義することはできない。

【0776】あるクラスによって定義されるフィーチャは、そのクラスのクラス定義、クラス”Class Definition”のメンバ、によって定義される。クラス自身は、クラスオブジェクト、クラス”class”のメンバ、によって表される。クラスは、識別子、クラス”Identifire”のメンバ、によって参照される。

【0777】クラスは、アペンディクスAに構造が説明されているクラス定義を最初に作りだすことによって作りだされる。クラス定義は、関係するクラスオブジェクトを作りだすためにオペレーション”makeClasses”を遂行するように指示される。オペレーション”makeClasses”のシンタックス及び挙動はアペンディクスAに一層詳細に説明されている。形成されたクラスオブジェクトの特別な形式及び構造は、(i) クラス定義オブジェクトに含まれる情報が保存され、(ii) サイテーション4108 (図96)、インターフェイスオブジェクト4110及びインプリメンテーションオブジェクト4112がクラスオブジェクトから導出される場合には重要ではない。条件(ii)は、アペンディクスBに説明するように、クラスオブジェクトの符号化にとって必要である。

【0778】一実施例において、クラス定義4100 (図96)のためにオペレーション”makeClasses”を遂行しているエンジンは、クラス定義4100のプロパティをクラスオブジェクト4102において複製することによって、クラスオブジェクト4102 (図97)を形成する。

【0779】オペレーション”makeClasses”を遂行したのちに、クラス定義4100は廃棄される。クラス定義4100のプロパティについては以下に一層詳細に説明する。クラス定義4100のプロパティはクラスオブジェクト4102において再生されるので、クラスオブジェクト4102は、クラス定義オブジェクト4100に含まれるすべての情報を含み、クラス定義オブジェクト4100のプロパティを導出することができる。

【0780】別の実施例において、コンパイルされたコンピュータプロセスの処理効率、サイテーション41

08、インターフェイスオブジェクト4110及びインプリメンテーションオブジェクト4112の情報を複製し、クラスオブジェクト4104内にサイテーション構造4108C (図98)、インターフェイス構造4110C及びインプリメンテーション構造4112Cをそれぞれ形成することによって実現される。解釈されるというよりはむしろコンパイルされるコンピュータ命令から成るコンピュータプロセスは、その命令が解釈されるコンピュータプロセスに比べて一般により効率的であり、すなわち、CPUクロックサイクルのより少ない数の範囲内である与えられたタスクを遂行する。前述したようにコンピュータ命令は最初は人が理解しうるソースコードである。コンピュータ命令がコンピュータによって実行されうる前に、この命令は、コンピュータによって理解されうるオブジェクトコード又はマシンコードに翻訳されなければならない。解釈されたコンピュータプロセスは各々の実行されるコンピュータ命令についてこのような翻訳を必要とし、コンパイルされたコンピュータプロセスは、コンパイルされたコンピュータプロセスの実行の間にこのような翻訳を必要としないので、コンパイルされたコンピュータプロセスは解釈されたコンピュータプロセスに比べて実質的により効率的であることができる。

【0781】サイテーション構造4108C、インターフェイス構造4110C及びインプリメンテーション構造4112Cは、サイテーション4108、インターフェイスオブジェクト4110及びインプリメンテーションオブジェクト4112に類似し、コンパイルされたコンピュータ命令、たとえば集散的にC++プログラミング言語を形成するコンピュータ命令から成っているオブジェクトである。

【0782】それぞれサイテーション構造4108C、インターフェイス構造4110C及びインプリメンテーション構造4112Cとして複製することによって、クラスオブジェクト4104によって表されるクラスに関する情報は、この情報をクラスオブジェクト4102

(図97)のサイテーション4108、インターフェイスオブジェクト4110及びインプリメンテーションオブジェクト4112からの情報を再生する場合に比べて一層効率的に検索することができる。サイテーション構造4108C (図98)、インターフェイス構造4110C及びインプリメンテーション構造4112Cに含まれる情報は、クラス定義4100 (図96)の対応するプロパティ内に含まれる情報と実質的に同等である。

【0783】クラスオブジェクト内に含まれる情報は、そのクラスオブジェクトがそれから導出されたクラス定義内に含まれる情報と実質的に同等であるので、クラス定義の構造の説明によって、クラスオブジェクトに含まれる情報の性質も同時に説明されることになる。

【0784】クラス定義オブジェクト

あるクラスによって定義されたフィーチャは、そのクラスを規定するクラス定義オブジェクトにおいて規定される。クラス定義オブジェクト4100の構造は図96に示されている。クラス定義オブジェクト4100は、プロパティ"citation"、"interface"、及び"implementation"を含む。サイテーション4108は、クラス定義オブジェクト4100のプロパティ"Citation"であり、クラス定義オブジェクト4100によって定義されたクラスを特定化するとともに、そのクラスのオーソリティ、タイトル及びエディションも特定する。あるサイテーションの構造については、以下にそしてアペンディクスAに一層詳細に説明される。インターフェイスオブジェクト4110及びインプリメンテーションオブジェクト4112は、それぞれクラス定義オブジェクト4100のプロパティ"interface"及び"implementation"である。

【0785】インターフェイスオブジェクト4110は、クラス定義オブジェクト4100によって定義されたフィーチャのインターフェイスを定義し、インプリメンテーションオブジェクト4112は、クラス定義オブジェクト4100によって定義されたフィーチャのインプリメンテーションを規定する。

【0786】インターフェイスオブジェクト  
インターフェイスオブジェクト4110の構造は図99に示されている。インターフェイスオブジェクト4110のプロパティには、プロパティ"isAbstract"、"classFeatures"、"sealedClassFeatures"、"instanceFeatures"、"sealedInstanceFeatures"、"vocabulary"及び"superclasses"が含まれる。

【0787】ブーリアン4208は、インターフェイスオブジェクト4110のプロパティ"isAbstract"である。ブーリアン4208が"true"であれば、インターフェイスオブジェクト4110をインターフェイスとするクラスは、抽象的である。さもなければクラスは具体的である。

【0788】レキシコン4210はインターフェイスオブジェクト4110のプロパティ"classFeatures"である。レキシコン4210は、そのクラスの"classFeatures"の定義のレキシコンである。あるクラスのクラスフィーチャは、クラス自身すなわちクラス定義オブジェクト4100（図96）によって表されるクラスがレスポндаとなるフィーチャである。レキシコン4210（図99）のキーたとえば、識別子4210Kは、クラスフィーチャをリファレンスする識別子である。レキシコン4210たとえばフィーチャ定義4210Vの値は、クラスフィーチャを定義するフィーチャ定義である。フィーチャ定義については以下に一層詳細に説明する。

【0789】セット4212は、インターフェイスオブジェクト4110のプロパティ"sealedClassFeatures

"である。セット4212は、シールされたクラスフィーチャをリファレンスする識別子たとえば識別子4212Iのセットである。シールされたフィーチャは、サブクラスが適合に対して阻止されるフィーチャである。あるフィーチャがシールされていない場合、クラス定義オブジェクト4100（図96）によって表されたクラスのサブクラスは、フィーチャのインプリメンテーションを再定義することによってそのフィーチャに適合することができる。

【0790】レキシコン4214（図99）は、インターフェイスオブジェクト4110のプロパティ"instanceFeatures"である。

【0791】レキシコン4214は、そのクラスの"instanceFeatures"の定義のレキシコンである。あるクラスのインスタンスフィーチャは、そのクラスのメンバがレスポндаであるフィーチャである。レキシコン4214（図99）のキー（レキシコン4214のキー又は値は示されていない）は、インスタンスフィーチャを参照する識別子であり、レキシコン4214の値は、インスタンスフィーチャを定義するフィーチャ定義である。

【0792】セット4216は、インターフェイスオブジェクト4110のプロパティ"sealedInstanceFeatures"である。セット4216は、シールされたインスタンスフィーチャをリファレンスする識別子（図示しない）のセットである。

【0793】レキシコン4218は、インターフェイスオブジェクト4110のプロパティ"vocabulary"である。レキシコン4218は、インターフェイスオブジェクト4110においてリファレンスされたユーザー定義されるクラスの識別子を、これらのクラスのサイテーションと関連させる。レキシコン4218のキーは、識別子、たとえば識別子4218Kであり、レキシコン4218の値は、対応するサイテーションたとえばサイテーション4218Vである。サイテーションについては以下にそしてアペンディクスAに一層詳細に説明される。インターフェイスオブジェクト4110すなわちレキシコン4218のプロパティ"vocabulary"は、インターフェイスオブジェクト4110及びクラス定義オブジェクト4100（図96）によって定義されたクラスの"vocabulary"である。

【0794】リスト4220（図99）は、インターフェイスオブジェクト4110のプロパティ"superclasses"である。リスト4220は、クラスをリファレンスする識別子たとえば識別子4220Iのリストである。リスト4220の識別子によってリファレンスされるクラスは、クラス定義オブジェクト4100（図96）によって定義されたクラスのイメディエイトインターフェイススーパークラスである。リスト4220（図99）の最後のアイテムを除く全てのアイテムは、ミックスインクラスを特定化し、リスト4220の最後のアイテム

は、フレーバを特定化する。フレーバを特定化するリスト4220の最後のアイテムの重要性については、フィーチャを実行する方法オブジェクトの選定及びオブジェクトのイニシャライゼーションに関連して以下に一層詳細に説明する。

#### 【0795】フィーチャ定義

フィーチャ定義すなわちクラス”Feature”のメンバは、ある特別のフィーチャのインターフェイスを定義する1つのオブジェクトである。フィーチャ定義4210V（図100）のプロパティには、プロパティ”例外”及び”isPublic”が含まれる。クラス”Feature”は抽象的であるので、フィーチャ定義4210Vは、クラス”Feature”のインスタンスではなく、そのメンバであり、オペレーション定義又は属性定義のどちらかである。これらの定義はどちらも以下にそしてアペンディクスAに説明される。セット4302は、フィーチャ定義4210Vのプロパティ”例外”である。セット4302は、定義されたフィーチャによって表明された例外がメンバであるクラスを各々リファレンスする識別子たとえば識別子4302Iである。フィーチャ定義4210Vによって定義されたフィーチャが例外を表明する場合、その例外は、セット4302の識別子によってリファレンスされたクラスの1つのもののメンバであることが確かめられる。例外がセット4302の識別子によってリファレンスされたクラスのメンバでない場合には、例外の代わりに、クラス”予期されない例外”のメンバが表明される。その他の場合には表明される。

【0796】ブーリアン4304は、フィーチャ定義の4210Vのプロパティ”isPublic”である。ブーリアン4304は、フィーチャ4210Vによって定義されたフィーチャがパブリックなフィーチャか、又はプライベートなフィーチャかを、すなわち、定義されたフィーチャのアクセスが”パブリック”であるか”プライベート”であるかを指示する。各々のフィーチャは、それに組み合わせられた”アクセス”を持ち、このアクセスは、どんなオブジェクトが、またどんな状況の下で各々のフィーチャが、適切にリクエストされうるかを定める。この発明の各々のフィーチャのアクセスについてはアペンディクスA、Bに示されている。

【0797】ブーリアン4304が”true”であれば、そのフィーチャはパブリックである。そのアクセスがパブリックであるフィーチャは、どんなオブジェクトによってリクエストされてもよい。ブーリアン4304が”false”である場合、そのフィーチャはプライベートである。そのアクセスがプライベートであるフィーチャは、レスポンスのみによってリクエストされる。すなわち、オブジェクトは、それ自身のフィーチャのみをリクエストすることができる。システムフィーチャすなわちそのアクセスが”システム”であるフィーチャは、すべてあらかじめ定義されており、したがってフィーチャ定

義によっては定義されない。

【0798】あらかじめ定義されたフィーチャは、エンジン内においてインプリメントされておりしたがってフィーチャ定義のようなオブジェクトによっては表されない。開示されたあらかじめ定義されたフィーチャをインプリメントする本発明の一実施例のエンジンの1つのインプリメンテーションは、全体として引用によって本明細書の一部となるマイクロフィルムアペンディクスGのソフトウェア中に開示されている。

#### 【0799】属性の定義

1つの属性を定義するフィーチャ定義は、アトリビュート（属性）の定義、すなわちクラス”Attribute”の1つのメンバである。したがってクラス”Attribute”はクラス”Feature”のサブクラスである。属性の定義はフィーチャの定義であるから、属性の定義は、前述したクラス”Feature”によって定義又は受け継がれたプロパティを引き継ぐ。さらに、属性の定義4400（図101）は、プロパティ”コンストレイント”及び”isSet”を含む。

【0800】コンストレイント4402は、属性定義4400のプロパティ”コンストレイント”である。コンストレイント4402は、属性定義4400によって定義された属性を表すオブジェクトを制約する。以下に一層詳細に説明するように、コンストレイントは、あるオブジェクトに対して拘束を設定し、（i）そのオブジェクトが1つのメンバであるクラス、（ii）オブジェクトがそのクラスのインスタンスでなければならないかどうか、（iii）オブジェクトがゼロでありうるか否か、及び（iv）フィーチャのリクエストとレスポンスとの間でオブジェクトがどのように通過されるか、を制限することができる。

【0801】ブーリアン4404は、属性定義4400のプロパティ”isSet”である。ブーリアン4404は、属性定義4400によって定義された属性が設定され質疑されうるか否かを指示する。ブーリアン4404が”true”であれば、属性定義4400は、設定可能な属性を規定する。そうでなければ、属性定義4400によって定義された属性は質疑のみされる。

#### 【0802】オペレーションの定義

前述したようにフィーチャは、属性であるか、又はオペレーションである。したがってクラス”Feature”の第2のサブクラスは、クラス”オペレーション”である。オペレーション定義すなわちクラス”オペレーション”のメンバは、あるオペレーションのインターフェイスを規定する。

【0803】図102は、オペレーション定義4500の構造を示す。オペレーション定義は、フィーチャ定義であるから、オペレーション定義4500は、前述したように、クラス”Feature”によって定義又は引き継がれるプロパティを含む。さらに、オペレーション定義

は、プロパティ"arguments"及び"result"を含む。リスト4502は、オペレーション定義4500のプロパティ"arguments"である。オペレーション定義のプロパティ"arguments"は、オプションであり、すなわち、あるオペレーション定義のプロパティ"arguments"は、リストでなく、ゼロであることができる。プロパティ"arguments"がゼロである場合、オペレーション定義4500によって定義されたオペレーションは、可変数のアーギュメントを消費する。アーギュメントの数が可変であるオペレーションの遂行についてはアペンディクスAに説明されている。逆に、プロパティ"arguments"がリスト4502であれば、オペレーション定義4500によって定義されたオペレーションによって消費されるアーギュメントの数は、ある固定された数のアーギュメントを消費し、この数は、リスト4502の長さに等しい。プロパティ"arguments"がエンブティリストであれば、オペレーションはいかなるアーギュメントも消費しない。リスト4502のアイテムたとえばコンストレイント4502Iは、オペレーション定義4500によって定義されたオペレーションのアーギュメントを制約するコンストレイントである。図103について以下に説明するように、コンストレイントは、オブジェクトのクラス及び通路を制約する。

【0804】コンストレイント4504は、オペレーション定義4500のプロパティ"result"である。コンストレイント4504は、オペレーション定義4500によって定義されたオペレーションの結果を制約する。オペレーション定義のプロパティ"result"もオプションである。

【0805】プロパティ"result"がコンストレイント4504でなくゼロであれば、オペレーション定義4500によって定義されたオペレーションはいかなる結果も生じない。

#### 【0806】コンストレイント

前述したように、コンストレイントは、あるオブジェクトのクラス及び通路を制約する。コンストレイント4402の構造については図103に示されている。コンストレイント4402のプロパティには、プロパティ"classID"、"ofClass"、"isInstance"、"isOptional"、及び"passage"が含まれる。

【0807】プロパティ"ofClass"及び"classID"は、制約されたオブジェクトがそのメンバでなければならないクラスを特定するための別の方法である。クラスオブジェクト4602は、コンストレイント4402のプロパティ"ofClass"である。クラスオブジェクト4602は、コンストレイント4402が設定されるオブジェクトをメンバとするべきクラスを表している。プロパティ"ofClass"はオプションであるため、ゼロであることができる。プロパティ"ofClass"がゼロである場合、そのオブジェクトをメンバとするべきクラスは、

コンストレイント4402のプロパティ"classID"である識別子4610によって参照される。

【0808】プロパティ"ofClass"は、クラスオブジェクトであるため、コンストレイント4402が作りだされた時において存在するクラスでしかありえない。しかし、あるコンストレイントのプロパティが、現存するクラスであることを要求した場合、本発明のユーザーは、複数の独立したクラスを定義することができなくなる。例えば、第1のクラスのフィーチャ定義においてのあるコンストレイントが、第2のクラスにおいての第2のフィーチャ定義の第2のコンストレイントが、第1のクラスを参照していることがあり得る。プロパティ"ofClass"に対する代替としてコンストレイント4402のプロパティ"classID"を用意すると、ユーザーは、複数の独立したクラスを同時に定義することができる。

【0809】ブーリアン4604は、コンストレイント4402のプロパティ"イズインスタンス"である。ブーリアン4604は、コンストレイント4402の設定されたオブジェクトがクラスオブジェクト4602によって現されるクラスのインスタンスでなければならないか否かを指示する。ブーリアン4604が"true"であれば、コンストレイントされたオブジェクトは、クラスの1つのインスタンスでなければならない。さもなければ、そのクラスにおいて、コンストレイントされたオブジェクトのメンバは、コンストレイント4402を満たすには不足する。

【0810】ブーリアン4606は、コンストレイント4402のプロパティ"イズオプション"である。ブーリアン4606は、プロパティ"ofClass"及び"classID"にもかかわらず、ゼロオブジェクトがコンストレイント4402を満たすか否かを指示する。ブーリアン4606が"true"であれば、ゼロオブジェクトはコンストレイント4402を満たす。本明細書中の多くの例において示すように、ゼロオブジェクトは、割愛されたオプションのアーギュメント又はあるフィーチャの結果の代わりにスタック中に置くことができる。

【0811】識別子4610はコンストレイント4402のプロパティ"passage"である。識別子4610は、コンストレイント4402が設定されたオブジェクトがどんな仕方でパラメーターとして、すなわちアーギュメント又は結果として通過するかを示している。識別子4610は、4つの可能な値、すなわち"byRef"、"byUnprotectedRef"、"byProtectedRef"及び"byCopy"を有する。識別子4610をこれら4つのうちの1つ以外の値に設定すると、クラス"Passage Invalid"のメンバである例外が表明される。

【0812】あるフィーチャのリクエストとそのフィーチャのレスポндаとの間のアーギュメントへの参照を通過させる際に、エンジンは、リクエストからのアーギュメントへのソースの参照を取り、レスポндаへの目的点



参照を供与する。あるレファランスを結果に導く際に、ソースレファランスはレスポンスから取られ、目的点レファランスはリクエストに供給される。

【0813】パラメーターが通過した”byRef”であると、ソースレファランスとデスティネーションレファランスとは互いに同一である。パラメーターが通過した”byUnprotectedRef”であると、ソースレファランスがされた保護されたレファランスである場合にクラス”レファランス保護”の例外が表明され、他の場合には、目的点レファランスはソースレファランスである。パラメーターが通過した”byProtectedRef”であると、目的点レファランスは、ソースレファランスによって参照されたオブジェクトへのプロテクトされたレファランスである。

【0814】パラメーターが通過した”byCopy”であると、目的点レファランスは、ソースレファランスによって参照されたオブジェクトのコピーへの保護されないレファランスである。パラメーターの通過についてはアペンディックスAに一層詳細に説明されている。

【0815】したがってコンストレイント4402（図101）は、オブジェクトのクラス及び通路を制限する。

【0816】インプリメンテーションオブジェクト  
前述したように、インプリメントオブジェクト4112（図96）は、クラス定義オブジェクト4100によって定義されたクラスインプリメンテーションを定義する。クラスインプリメンテーションは次のものを定義する。(i) 定義されたクラスのメンバのイチジョウのプロパティ。(ii) いろいろなクラスフィーチャのそれぞれのインプリメンテーション。(iii) あるクラスによって定義されたインスタンスフィーチャ。(iv) クラスについて定義された属性を設定する方法。(v) オブジェクトを規定されたクラスから第2のクラスに変換する方法。(vi) オブジェクトを第2のクラスから規定されたクラスに変換する方法。(vii) クラスの定義に用いられる識別子の定義。(viii) 定義されたクラスのインプリメンテーションサブスーパークラス。

【0817】リスト4702（図104）は、インプリメンテーションオブジェクト4112のプロパティ”properties”である。リスト4702は、インプリメンテーションオブジェクト4112がインプリメントするプロパティを定義する。レキシコン4702のアイテム例えば識別子4702Iは、インプリメンテーションオブジェクトによって規定されたプロパティの識別子である。

【0818】レキシコン4704（図104）は、インプリメンテーションオブジェクト4112のプロパティ”classMethods”である。レキシコン4704は、クラス定義オブジェクト4100（図96）によって現されたクラスのクラスフィーチャのための方法を提供す

る。レキシコン4704のキー、例えば識別子4704K（図104）は、クラスフィーチャの識別子であり、レキシコン4704の値、例えば方法オブジェクト4704Vは、クラスフィーチャのインプリメンテーションを規定する方法オブジェクトである。

【0819】方法オブジェクトすなわちクラス”方法”のメンバは、フィーチャのメソッドを規定するオブジェクトである。あるフィーチャの方法は、そのフィーチャの遂行にあたってエンジン、したがってそのエンジンが実行されているコンピューターシステムが行う特定のステップ連鎖である。

【0820】ある方法は、開示された命令セットがユーザー定義されるクラス又はフィーチャのどれかによって申請される場合に、その開示された命令セットの言葉で定義される。方法オブジェクトは以下に、及びアペンディックスAに、一層詳細に説明されている。

【0821】レキシコン4706は、インプリメンテーションオブジェクト4112のプロパティ”インスタンスメソッド”である。レキシコン4706はインスタンスフィーチャの識別子に対応する方法オブジェクトに関連させ、対応するこれらの方法オブジェクトは、インスタンスフィーチャのインプリメンテーションを規定する。レキシコン4706の構造は、レキシコン4704の構造と同様であり、この説明は引用によって本明細書の一部となる。

【0822】レキシコン4707は、インプリメンテーションオブジェクト4112のプロパティ”セットメソッド”である。レキシコン4707は、それぞれの属性を特定する識別子を方法オブジェクトと関連させる。レキシコン4707の方法オブジェクトの各々は、”セット”モディファイヤーの存在下に行われたときに対応する識別子の属性をインプリメントする。換言すれば、レキシコン4707の各々の方法オブジェクトは、対応する属性の設定のインプリメンテーションを規定する。”セット”モディファイヤーの存在下に属性を実行することについてはアペンディックスAに説明されている。

【0823】レキシコン4708は、インプリメンテーションオブジェクト4112のプロパティ”フロムメソッド”である。レキシコン4708は、識別子を方法オブジェクトと関連させる。各々の識別子は、あるクラスを参照し、方法オブジェクトに組み合わされる。この方法オブジェクトは、レファランスされたクラスから、クラス定義オブジェクト4100（図96）すなわちそのインプリメンテーションのインプリメンテーションオブジェクト4112（図104）である。クラスによって定義されたクラスへの変換のインプリメンテーションを提供する。

【0824】レキシコン4710は、インプリメンテーションオブジェクト4112のプロパティ”ツーマソッ

ズ”である。レキシコン4710は、識別子を方法オブジェクトと組み合わせる。各々の識別子は、クラスを参照し、方法オブジェクトと関連される。この方法オブジェクトは、クラス規定オブジェクト4100（図96）によって規定されたクラス、すなわちそのインプリメンテーションがインプリメンテーションオブジェクト4112（図104）であるクラスからの参照されたクラスの変換のインプリメンテーションを行う。レキシコン4707、4708及び4710の構造は、レキシコン4704の構造と同様であり、この説明は引用によってこの明細書の一部分となる。

【0825】レキシコン4712は、インプリメンテーションオブジェクト4112のプロパティ“vocabulary”である。レキシコン4712は、クラスを特定するためにインプリメンテーションオブジェクト4112によって使用される識別子を定義する。レキシコン4712のキー例えば識別子4712Kは、特定のユーザー定義されたクラスを特定するためにインプリメンテーションオブジェクト4112内において用いられる識別子である。レキシコン4712の値、例えばサイテーション4712Vは、関連した識別子によって特定されるクラスを対応するサイテーションである。レキシコン4712は、インプリメンテーションオブジェクト4112中のユーザー定義されたクラスを特定するために用いられる識別子とネットワークを通じてクラスを特定するために用いられるサイテーションとの間の翻訳を提供する。サイテーションについては以下にまたアペンディックスAに詳細に説明されている。

【0826】リスト4714は、インプリメンテーションオブジェクト4112のプロパティ“superclasses”である。リスト4714のアイテム、例えば識別子4714Iは、クラス定義4100（図96）によって定義されたクラスインプリメンテーションスーパークラスを特定する識別子である。リスト4714（図104）は、すなわちインプリメンテーションオブジェクト4112のプロパティ“superclasses”は、リスト4220のアイテムでないクラスの識別子、すなわちインターフェースオブジェクト4110のプロパティ“superclasses”を含みうる。一例として第1のクラスは、第1のクラスのフィーチャのインプリメンテーションにおいて、第2のクラスのフィーチャを使用する能力によって有利となりうる。

【0827】しかし、このようなクラスをデザインする場合、第2のクラスのフィーチャ自身が第1のクラスから受け継がれることは必ずしも好ましくない。このような場合、第2のクラスは、インプリメンテーションスーパークラスとされるが、第1のクラスのインターフェーススーパークラスとはされない。

【0828】したがって、インプリメンテーションオブジェクト4112は、クラス定義オブジェクト4110

（図96）によって規定されるクラスについて、(a) 規定されたクラスのメンバのプロパティ、(b) クラス及びインスタンスのフィーチャ方法、(c) 属性を設定する方法、(d) 定義されたクラスへの、又はその定義されたクラスへの変換方法、(e) クラスを特定するために用いられる識別子及び(f) インプリメンテーションスーパークラス、を定義する。

#### 【0829】方法オブジェクト

前述したように、方法オブジェクトは、フィーチャ又は変換を遂行する際に実行されるコンピューターの命令を規定する。図105は方法オブジェクト4800の構造を示す。方法オブジェクト4800のプロパティは、プロパティ“procedure”及び“変数”を含む。プロシージャ4802は、方法オブジェクト4800のプロパティ“procedure”である。プロシージャ4802の遂行は、方法オブジェクト4800によってインプリメントされたフィーチャ又は変換の実行を構成する。プロシージャについては以下に詳細に説明する。

【0830】リスト4804は、方法オブジェクト4800のプロパティ“変数”である。リスト4804は、プロシージャ4802によって使用される変数を規定する。リスト4804のアイテム、例えば識別子4804Iは、方法オブジェクト4800について規定された変数をレファランスする識別子である。ある方法の変数の値を示す関連されたオブジェクトは、その方法の遂行の間、ユーザー定義されたフレームのプロパティ“変数”としてリスト中に確認される。フレームについては以下にそしてアペンディックスAに詳細に説明されている。

#### 【0831】プロシージャ

プロシージャ4802の構造は図106に示される。

【0832】プロシージャ4802は、オブジェクト4902A-4902Eのリスト4902を含む。リスト4902は、クラス“List”のメンバではなく、その代わり、プロシージャ4802の一体的な部分である。換言すれば、リスト4902は、プロシージャ4802のプロパティ又はアイテムではなく、リスト4902はクラス“List”のメンバとして取り扱われることを供与するいかなるフィーチャも用意されていない。本発明の一実施例によれば、クラス“procedure”はクラス“List”のインプリメンテーションサブクラスであるが、クラス“List”のインターフェースサブクラスでない。したがって、クラス“List”によって規定されたフィーチャは、クラス“procedure”によって受け継がれず、クラス“List”によって規定された方法は、リスト4902をつくり出して、それにアクセスするプロシージャによって使用することができる。

【0833】オブジェクト4902A-4902Eは、実行されるオブジェクト、すなわち、ミックスインクラス“実行”のメンバである。プロシージャ4802は、順次すなわちオブジェクト4902A、オブジェクト4

902B、オブジェクト4902C、オブジェクト4902D及びオブジェクト4902Eの順序でオブジェクト4902A-4902Eを実行することによって遂行される。プロシージャ4802の実行の間に、予め規定されたフレームは、オブジェクト4902A-4902Eのうちどれが実行されているかを決めて、プロシージャ4802の遂行の動的な状態を記録し保持する。予め規定されたフレームについては以下に、及びアペンディックスBに、詳細に説明されている。

#### 【0834】サイテーション

サイテーションは、各々後ろ方向又は前方向に互いにコンパティブルである一連のオブジェクトを特定するオブジェクトである。サイテーションはまた、シリーズ中の特定のオブジェクト及びプロセス又はその特別なオブジェクトを作り出したプロセスのオーソリティをさらに特定する。既に説明したように、クラスはサイト（引用）されたオブジェクトである。したがって、サイテーションは、互いに後ろ方向及び前方向にコンパティブルな一連のクラスを特定するために用いられる。

【0835】後ろ方向及び前方向のコンパティブルについては以下に、及びアペンディックスAに、詳細に説明される。

【0836】なおそのサイテーションが互いに前方向又は後ろ方向コンパティブルとしてオブジェクトを識別するオブジェクトは、必ずしも前方向又は後ろ方向にコンパティブルではない。一例として、それぞれのサイテーションによってコンパティブルであると特定された2つのクラス定義オブジェクトは単にコンパティブルであることが意図されているに過ぎない。

【0837】サイテーション5000（図107）はクラス"Citation"の1つのメンバである。サイテーション5000のプロパティには、プロパティ"タイトル"、"メジャーエディション"、"マイナーエディション"、及び"オーサー"が含まれる。サイテーション5000のプロパティ"タイトル"は識別子5008である。したがって識別子5008はサイテーション5000のタイトルである。サイテーションのタイトルはサイテーションによって現される連鎖をレファランす。サイテーションのタイトルは、サイテーションのオーソリティに関して解釈される。換言すれば、異なったオーソリティの2つのサイテーションは同等のタイトルを使用することができる。2つのサイテーションは、異なったオーソリティによって双方から識別される。サイテーションのオーソリティは、以下に説明するプロパティ"オーサー"によって特定化される。

【0838】プロパティ"メジャーエディション"及び"マイナーエディション"は、どちらも整数5004、5006である。したがって、整数5004、5006は、タイトルサイテーション5000のメジャーエディション及びマイナーエディションである。ある連

鎖中の2つのオブジェクトの間の関係は、それらのオブジェクトの相対的なメジャーエディション及びマイナーエディションによって定められる。一例として、オブジェクト5102、5104（図108）は、それぞれサイテーション5106、5108によって引用される。整数5110、5112は、サイテーション5106のメジャーエディションとマイナーエディションとをそれぞれ現す。整数5114、5116は、サイテーション5108のそれぞれメジャーエディションとマイナーエディションとを現す。

【0839】サイテーション5106のメジャーエディションである整数5110が、サイテーション5108のマイナーエディションである整数5114よりも大きい場合には、オブジェクト5102は、オブジェクト5104の生成よりも後に生成されたものであり、オブジェクト5104に対して後ろ方向にコンパティブルである。

【0840】オブジェクト5102がオブジェクト5104に対して後ろ方向にコンパティブルであれば、オブジェクト5104は、オブジェクト5102に対して前方向にコンパティブルである。その逆に、整数5100が、整数5114よりも小さい場合には、オブジェクト5104は、オブジェクト5102の生成よりも後に生成され、オブジェクト5102に対して後ろ方向にコンパティブルである。

【0841】整数5110、5114が等しければ、それぞれオブジェクト5102、5104のマイナーエディションを表す整数5112、5116の相対値は、オブジェクト5102、5104の間の関係を定める。例えば、整数5112が整数5116よりも大きければ、オブジェクト5102はオブジェクト5104の生成よりも遅れて生成され、オブジェクト5104に関して後ろ方向にコンパティブルである。その逆に整数5112が整数5116よりも小さければ、オブジェクト5104はオブジェクト5102の生成よりも生成の後に生成され、オブジェクト5102に関して後ろ方向にコンパティブルである。

【0842】プロパティ"オーサー"はテレネーム5002（図107）である。したがって、テレネーム5002はサイテーション5000のオーサーである。テレネーム5002はサイテーション5000によって特定されるオブジェクトを生成したプロセスのネームである。単一の連鎖の全てのオブジェクトは、単一のオーソリティのプロセスによって作り出されている。

【0843】前方向及び後方向のコンパティビリティについてのより正確な定義はアペンディックスAに示されているが、次の例は、これらのコンセプトの意味及び有用性を示している。アペンディックスAに定義されるように、クラス"Integer"のメンバすなわち整数は、"加算"、"減算"、"掛け算"及び"割り算"を含むあ

る数の数学的な演算を実行することができる。クラス“Integer”のプロパティ“Citation”は、クラス“Integer”を特定化する第1のサイテーションである。本明細書中に記載されるコンピューター命令のセットをインプリメントし、また本明細書中に記載されるコンピューター命令のセットを使用する際に新しいクラス及び新しいフィーチャを定義する際に、アペンディックスAに記載されたクラス“Integer”の記述に適合するクラス“Integer”のメンバに依存して過度に多くのフィーチャが設計されている。

【0844】本明細書に記述されるコンピューター命令のセットの、以下のバージョンにおいて、乱数を生成させるためにオペレーション“randomize”を与えるように第2のクラス“Integer”が設計されているものと想定する。第1のクラス“Integer”は、第1のクラス“Integer”と呼ばれる。第2のクラス“Integer”のプロパティ“Citation”は、第2のサイテーションである。第1のクラス“Integer”と第2のクラス“Integer”との間には他の差異は存在せず、第1のクラス“Integer”のメンバについてリクエストされたいかなるフィーチャも第2のクラス“Integer”のメンバによって満たされているものと想定する。したがって、第2のクラス“Integer”の整数が与えられていれば、第1のクラス“Integer”の記述に適合した整数に依存して設計されたフィーチャは、適切に機能する。したがって、第2のクラス“Integer”は、第1のクラス“Integer”と後ろ方向にコンパティブルである。その逆に、第1のクラス“Integer”は、第2のクラス“Integer”と前方向にコンパティブルである。

【0845】第1のクラスと第1及び第2のサイテーションのプロパティ“タイトル”は、等しく、どちらかのクラス“Integer”をレファランスする。しかし、第1及び第2のサイテーションのメジャーエディション及びマイナーエディションは、第1のクラス“Integer”と第2のクラス“Integer”との間の前方向及び後ろ方向のコンパティビリティを反映している。後ろ方向及び前方向にコンパティブルなオブジェクトがネットワーク内において共存することを許容すると、本明細書により記述されるコンピューター命令のセットの初期のバージョンによって作り出されたオブジェクトは、本明細中に記述されるコンピューター命令のセットの後のバージョンをインプリメントするエンジン内において移動し動作することはできる。以上の説明からわかるように、クラスはそれ自身に対して前方向及び後ろ方向にコンパティブルである。以上の例示的な実施例は、予め限定されたクラスのコンテキストにおいて前方向及び後ろ方向のコンパティビリティを示しているが、ユーザーによって定義されたクラスは、典型的に、予め規定されたクラスよりもより頻繁に変化する。そのため、クラスの間の前方向及び後ろ方向のコンパティビリティを与えることは、

ユーザーによって定義されたクラスのコンテキストにおいては特に重要である。

【0846】すなわち、クラスは、ここに開示された命令セットの一部となり、したがって、可搬型であり、エージェントとともに第1のコンピューターシステムから第2のコンピューターシステムに移動することができる。

#### 【0847】解釈される命令セット

前述したように、本発明の3つの要素によって、コンピュータープロセスは特に可動となりかつ一般的になる。第1に本発明のコンピューター命令セットは、均質又は異質なネットワーク内において均質にインプリメントされている。第2に、クラスは、開示されたコンピューター命令セットのオブジェクトであるため、命令セットは拡張性でありかつ可動である。すなわち遠隔なコンピューターシステムにおいて定義されていないクラスは、可動のコンピュータープロセスとともにこれらの変革のコンピューターシステムに転送される。以下の説明は、コンピューターの命令セットを特に一般的に可動にする第3の要素に関係している。ここに開示されたコンピューター命令セットのコンピューター命令は解釈される。

【0848】多くのコンピュータープロセスは最初に人が読むことができるソースコードで書かれ、次にこのソースコードがオブジェクトコードにコンパイルされ、マシンコードにリンクされる。このマシンコードは人には全く読むことができない。ソースコードをマシンコードに翻訳する主な利点は、マシンコードがソースコードを再コンパイルすることなく、何回でもコンピューターシステムのCPUによって実行されるため、翻訳を一度だけ行えば良いことになる。しかし異質なネットワークの2つのコンピューターシステムは、同一のマシンコードの命令をともに実行しない。したがって第1のコンピューターシステムから第2のコンピューターシステムに向かって移動するエージェントが、第1のコンピューターシステムにも第2のコンピューターシステムにも特異でない標準化された命令セットによって現されることが好ましい。

【0849】ソースコード1回だけコンパイルする利点がネットワークの異質性によって除かれるため、開示されたコンピューター命令セットは解釈される。本明細書中において“解釈される”という用語は、当該技術において理解されている通りに用いられている。すなわち、一連のコンピューター命令中のコンピューター命令が読まれマシンコードに翻訳され、一連のコンピューター命令中の次にコンピューター命令が読まれる前にエンジンによって実行される。開示された命令セットの命令をコンパイルするのではなく解釈することによって一層大きな一般性が得られる。第1のエージェントは第1のコンピューターシステムから第2のコンピューターシステムに向かって移動しそこで、第1のエージェントにプロシー

ジャを与える第2のエージェントと出会う。第1のエージェントは次に、第1のエージェントが当初は成功するように設計されていなかったプロシージャを遂行する。

【0850】以下の説明は、ここに開示されたコンピューターの解釈についての説明である。以下の説明はプロシージャとその個々のアイテムの実行に主に向けられ、個々のアイテムは、個々のコンピューター命令である。

#### 【0851】プロシージャの実行

あるプロシージャは、オペレーション”do”をそのプロシージャが行うことをリクエストする命令を表出することによって遂行される。プロシージャの遂行を表示させる他のプロシージャは以下に、そしてアペンディックスAに説明されている。クラス”procedure”について定義されたオペレーション”do”の遂行は、フローチャート5200（図109）に示され、予め定義されたフレーム5250（図110）によって現されている。予め定義されたフレーム5250は、それぞれ整数5252及びプロシージャ4802であるプロパティ”position”及び”procedure”を含む。予め定義されたフレームはアペンディックスAに詳細に説明されている。予め定義されたプロシージャ5250は、プロシージャ4802（図106）によって遂行されるオペレーション”do”の動的な状態を記録する。

【0852】ステップ5202（図109）において、整数5252は、値1に設定される。

【0853】整数5252はその実行が最も最近に開始されたアイテムのプロシージャ4802内の位置を特定する。処理はステップ5202（図109）からエンドオブプロシージャテストステップ5204に移行し、ここで整数5252（図110）がプロシージャ4802の長さと比較される。整数5252は、プロシージャ4802の長さに比べて短い、又は等しい場合に処理はエンドオブプロシージャテストステップ5204（図109）からステップ5208に移行する。ステップ5208において、プロシージャ4802（図110）内の位置がインテジャー5252であるアイテムが遂行される。実行されるオブジェクトの遂行については、以下にそしてアペンディックスAに詳細に説明される。

【0854】プロシージャ4802のアイテムが遂行されると、処理はステップ5208（図109）から例外テストステップ5210に移行する。例外テストステップ5210において、エンジン解釈プロシージャ102（図110）は、そのアイテムの遂行が成功したか、又は例外を表明したかを定める。例外が表明された場合処理は例外テストステップ5210（図109）から終了ステップ5212に移行し、そこでオペレーション”do”が失敗に終わり、プロシージャ4802（図110）の遂行が終了する。

【0855】ステップ5208（図109）において整数5252により示されるプロシージャ4802内のう

ちでアイテムの遂行によって例外が表明された場合処理が例外テストステップ5210からステップ5214に移行し、ここで整数5252（図110）がインクリメントされる。処理はステップ5214（図109）からエンドオブプロシージャテストステップ5204に移行し、ここで予め定義されたフレーム5250（図110）のプロシージャ”position”すなわち整数5252が前述したように応答プロシージャ4802の長さと比較される。

【0856】整数5252（図110）がプロシージャ4802の長さよりも大きい場合、処理はエンドオブプロシージャテストステップ5204（図109）から終了ステップ5216に移行し、そこでオペレーション”do”は成功のうちに終了する。

【0857】このようにプロシージャ4802（図106及び図110）によってオペレーション”do”が遂行されることによって前述した順序でプロシージャ4802の各々のアイテムが実行される。

#### 【0858】実行モデル

前述したように、あるエンジン例えばエンジン132B（図34）の主な機能は、プロシージャの遂行すなわちプロシージャのアイテムの実行である。本発明のここに開示された実施例の実行モデルはプロシージャの各アイテムが実行されるプロセスである。ここに開示された実施例の実行モデルの説明は6つのセクションに分けられる。第1にフレームの構造及び組織が説明される。フレームは1つのフィーチャ又は1つの変更をインプリメントする方法の動的状態を記録するオブジェクトである。第2に識別子の実行が説明される。あるフィーチャにレファランスする識別子の実行は、レファランスされたフィーチャの遂行を生じさせる。第3に、あるフィーチャ又はコンバージョンをインプリメントする方法オブジェクトをクラスハイアラキーから選出するプロセスが説明される。方法オブジェクトを検索するクラスオブジェクトの構造は前述した通りである。第4にあるフィーチャのエスカレーションが説明される。第5にモディファイアーの実行が説明される。第6にセレクターの実行が説明される。

#### 【0859】フレーム

フレームは、フィーチャ又は変更の実行の間、フィーチャ又は変更をインプリメントする方法の動的状態を記録するオブジェクトである。あるプロセスすなわちエージェント又はブレイスの実行状態は、1以上のフレーム中に記録される。1つのプロセスが作り出されるとき、そのプロセスはオペレーション”live”を遂行するように求められる。プロセス5300内にはプロパティ”フレーム”が作り出され、このプロパティ”フレーム”は、プロセス5300の実行スレッドを現すスタック5302である。

【0860】スタック5302はプロセス5300の実

行状態を集散的に形成するフレームを収容している。プロセス5300が最初にオペレーション"live"を遂行するように求められたとき、スタック5302の単一のアイテムは、オペレーション"live"の動的状態を記録するユーザーによって定義されたフレーム5304である。

【0861】予め定義されたクラス"Process"、"Agent"、及び"Place"は抽象的であるので、本発明のここに開示された実施例のプロセスはユーザーによって定義されたクラスのインスタンスである。予め定義されたアブストラクトのクラス"Process"、"Agent"、及び"Place"はオペレーション"live"のインプリメンテーションを定義しない。したがって、オペレーション"live"の方法の動的状態を記録するフレーム5304は必然的にユーザーによって定義されたフレームである。ユーザーによって定義されたフレームは、オペレーションの実行の間ユーザーによって定義されたインプリメンテーションによってオペレーションの動的状態を記録するフレームである。ユーザーによって定義されたフレーム5304の構造は図112に示されている。

【0862】ユーザーによって定義されたフレーム5304（図112）のプロパティには、プロパティ"class"、"responder"、"procedure"、"position"、"stack"及び"variables"が含まれる。クラスオブジェクト5402は、ユーザーによって定義されたフレーム5304のプロパティ"class"であり、これはスーパークラス"Object"から受け継がれたプロパティである。クラスオブジェクト5402は、ユーザーによって定義されたフレーム5304が1つのインスタンスであるクラスを現し、したがってクラス"ユーザーによって定義されたフレーム"又はそのサブクラスを現す。

【0863】オブジェクト5404はユーザーによって定義されたフレーム5304のプロパティ"responder"である。オブジェクト5404はユーザーによって定義されたフレーム5304がその動的状態を記録するフィーチャ又は変更のレスポンドである。図111のコンテキストにおいてオブジェクト5404は、プロセス5300がオペレーション"live"のレスポンドであるため、プロセス5300である。

【0864】プロシージャ5406（図112）は、スーパークラス"Procedure Frame"から受け継がれたプロパティである。ユーザーによって定義されたフレーム5304のプロパティ"procedure"である。

【0865】プロシージャ5406は、その動的な状態をユーザーによって定義されたフレーム5304が記録するところの方法オブジェクトのプロパティ"procedure"である。一例として、ユーザーによって定義されたフレーム5304が方法オブジェクト4800（図105）の動的状態を記録する場合、プロシージャ480

2、5406は同一のプロシージャである。

【0866】整数5408は、スーパークラス"Procedure Frame"から受け継がれたユーザーによって定義されたフレーム5304のプロパティ"position"である。整数5408は、現在実行されているプロシージャ5406のアイテムのプロシージャ5406内のポジション(1)を特定する。一例としてプロシージャ5406の第2のアイテムが、その実行が開始されているがまだ終了していないプロシージャ5406のアイテムである場合、整数5408の値は2となる。

【0867】ユーザーによって定義されたフレーム5304は、2つの追加のプロパティ、すなわちプロパティ"stack"及び"変数"を有する。スタック5410は、ユーザーによって定義されたフレーム5304のプロパティ"stack"であり、ユーザーによって定義されたフレーム5304のスタックである。プロシージャ5406の実行の開始時に、スタック5410は、その動的状態がユーザーによって定義されたフレーム5304によって記録されたフィーチャ又は変更の遂行によって消費されたアーギュメントを有している。プロシージャ5406の遂行後には、スタック5410は、フィーチャ又は変更の遂行によって生じた結果（もしあれば）を有している。

【0868】リスト5412は、ユーザーによって定義されたフレーム5304のプロパティ"変数"である。リスト5412のアイテム、例えばオブジェクト5412Iは、フレームの変数である。変数はプロシージャ5406が遂行される間に、ユーザーによって定義されたフレーム5304の動的状態を記録する。リスト5412（図112）のアイテムである変数は、その動的状態がユーザーによって定義されたフレーム5304によって記録されている方法オブジェクトのプロパティ"変数"の同じ位置にあるアイテムによってレファランスされる。一例として、ユーザーによって定義されたフレーム5304が方法オブジェクト4800（図105）の動的状態を記録する場合、リスト4804内のポジション1にある識別子4804Iは、リスト5412内の1に有るオブジェクト5412I（図112）をレファランスする。

【0869】このように、ユーザーによって定義されたフレーム5304は、(i) フィーチャ又は変更のレスポンド、(ii) フィーチャ又はコンバージョンをインプリメントするプロシージャ、(iii) 現在実行されている命令されているプロシージャ内の位置、(iv) フィーチャ又は変更のアーギュメント又は結果を収容しているスタック、及び(v) ユーザーによって定義されたフレーム5304の動的状態を記録する変数を記録することによって、ユーザーによって定義されたフィーチャ又は変更の動的状態を記録する。

【0870】実行されるオブジェクトの実行

前述したように、プロシージャの遂行は、プロシージャの各アイテムの順次の実行である。一般に実行されるオブジェクトすなわちミックスインクラス”実行”から受け継がれたオブジェクトの実行は、そのオブジェクトに対するレファランスが、プロシージャの実行状態を現すフレームのスタック、例えばスタック5410に保存される結果となる。しかし、識別子、モディファイアー及びセクターの実行は、この一般的な規則に対する例外となる。

#### 【0871】識別子

他の指示をしているモディファイアーの存在の元の実行されるのでない限り、実行される識別子は、あるフィーチャをレファランスしているものと想定され、したがってあるオペレーションを呼び出すか、又はある属性を質疑している。第1のプロシージャ内のある識別子の実行は、その識別子によってレファランスされたフィーチャを遂行させる結果となり、それによって第2のプロシージャは、識別子によってレファランスされたオペレーション又は属性をインプリメントする方法オブジェクトのプロシージャである。フィーチャをレファランスする識別子の実行は、フローチャート5500（図55）に示されている。

【0872】フローチャート5500は、図115～図117のコンテキストにおいて説明される。スタック5302（図115）は、プロセス5300の実行スレッドすなわちプロパティ”フレーム”である。

【0873】ユーザーによって定義されたフレーム5304は現在のフレームである。現在のフレームは、プロシージャすなわちそのアイテムが現在エンジンによって実行されているプロシージャ5406を含むフレームである。換言すれば、プロセス5304の解釈においてエンジンが実行する、次に実行されるオブジェクトは、ユーザーによって定義されたフレーム5304のプロシージャから検索される。

【0874】プロシージャ5406はユーザーによって定義されたフレーム5304のプロパティ”procedure”である。整数5408は、ユーザーによって定義されたフレーム5304のプロパティ”position”であり、その実行が最も最近にエンジン処理プロセス5300によって開始されたアイテムのプロシージャ5406内の位置を示している。識別子の実行は、整数5408によって示される位置にあるプロシージャ5406のアイテムが識別子である場合に生じる。この例では識別子はあるオペレーションを参照（レファランスする）する。識別子によってレファランスされるオペレーションは、図115～図117及び図55のコンテキストにおいて、時にオペレーション”subject”と呼ばれる。識別子の実行は、フローチャート5500（図55）に示される。リクエストのスタックすなわちスタック5410は、ステップ5502においてユーザーによって定義

されたフレーム5304のアトリビュート”stack”に質疑することによって生ずる。ユーザーによって定義されたフレーム5304のプロシージャ5405（図115）の遂行が、整数5408によって示されたプロシージャ5406内の位置に有る識別子の実行をリクエストするので、現在のフレームである、ユーザーによって定義されたフレーム5304のプロパティ”responder”は、識別子によってレファランスされるオペレーションのリクエストである。図115のコンテキストにおいてレスポンドはプロセス5300である。したがってスタック5410は、”リクエストのスタック”である。

【0875】処理はステップ5502（図55）からスタックエンプティテストステップ5504に移行する。識別子が実行される時点において、スタック5410の上部にあるオブジェクトは、サブジェクトオペレーションのレスポンドである。スタックエンプティテストステップ5504において、エンジン解釈プロセス5300は、スタック5410がエンプティであるか否かを定める。

【0876】スタック5410がエンプティであることが定められたら、処理は、スタックエンプティテストステップ5504から終了ステップ5506に移行し、ここでクラス”Responder Missing”の例外が表明され、サブジェクトオペレーションは失敗に終わる。しかし、スタック5410がエンプティでない場合、処理はスタックエンプティテストステップからステップ5508に移行し、ここで、スタック5410の上部にあるオブジェクト（図示しない）はスタック5410からポップ（復元）される。スタック5410からポップされるオブジェクトは、実行中の識別子のサブジェクトオペレーションの予定されたレスポンドであり、時には、図115～図117及び図108、109のコンテキストにおいて、”レスポンドオブジェクト”と呼ばれる。

【0877】なお、サブジェクトオペレーションは、ユーザーによって定義され、したがってユーザーによって定義されたフレームによって現される。予め定義されたフィーチャは直接にエンジン中表示される。換言すれば、予め定義されたオペレーションを形成する命令は、エンジン中にあるコンピュータープロセスを形成する命令に含まれている。実行されるアイデンティファイアーが予め定義されたオペレーションをレファランスする場合、エンジン内のオペレーションを定義する命令は、突き止められ、遂行される。

【0878】本明細書中及びアペンディックスAに記述される予め定義されたオペレーションを規定する命令を含む、本発明の一実施例のエンジンの1つのインプリメンテーションは、本明細書に全体として引用によって合体されるマイクロフィルムアペンディックスGのソフトウェアに開示されている。

【0879】処理はステップ5508からステップ55



10に移行する。ステップ5510では、サブジェクトオペレーションを定義する定義オペレーションが検索される。定義オペレーション及びそれに関連した方法オブジェクトの選択については、以下に詳細に説明する。処理はステップ5510からアクセステストステップ5512に移行する。アクセステストステップ5512では、サブジェクトオペレーションのアクセスが検査される。プロパティ“イズパブリック”、例えばブーリアン4404(図101)は、定義オペレーションから検索される。サブジェクトオペレーションのアクセスが“パブリック”でない場合、すなわちプロパティ“イズパブリック”が“false”であり、レスポndingオブジェクト(応答オブジェクト)がユーザーによって定義されたフレーム5304のプロパティ“responder”(図示しない)でない場合、処理はアクセステストステップ5512から終了テストステップ5514に移行する。

【0880】終了ステップ5514では、クラス“Feature Unavailable”の例外が表明され、サブジェクトオペレーションは失敗に終わる。しかしプロパティ“イズパブリック”が“true”であるか、又はレスポndingオブジェクトがユーザーによって定義されたフレーム5304のプロパティ“responder”である場合には、サブジェクトオペレーションのアクセスは容易に可能であり、処理はアクセステストステップ5512からアーギュメント数テストステップ5516に移行する。

【0881】アーギュメント数テストステップ5516においては、定義オペレーションのプロパティ“arguments”が、定義オペレーションのアトリビュート“arguments”の質疑によって生成される。プロパティ“arguments”のリスト中のアイテム数は、スタック5410上のオブジェクトの数と比較される。プロパティ“arguments”がスタック5410上に存在するオブジェクトの数よりも大きなアーギュメントの数を規定した場合には、処理はアーギュメント数テストステップ5516から終了ステップ5518に移行する。終了ステップ5518においてはクラス“Argument Missing”の例外が表明され、サブジェクトオペレーションは失敗に終わる。

【0882】なお、終了ステップ5506又は終了ステップ5514において、サブジェクトオペレーションにレファランスする識別子の実行によって表明された例外は、サブジェクトオペレーションを終了に終わらせ、同一の例外が表明される。

【0883】ステップ5510において検索された定義オペレーションのプロパティ“arguments”によって示されたアーギュメントと少なくとも同数のオブジェクトがスタック5410に存在していれば、処理はアーギュメント数テストステップ5516からアーギュメントクラステストステップ5520に移行する。アーギュメントクラステストステップ5520において、エンジン解釈プロセス5300は、ステップ5510において検索

された定義オペレーションのプロパティ“arguments”であるリスト内の対応するコンストレントをアーギュメントのポジションにおいてスタック5410の各々のオブジェクトが満たすか否かを定める。

【0884】一例として、定義オペレーション4500のプロパティ“arguments”であるリスト4502(図102)のコンストレント4502Iは、スタック5410の上部のオブジェクトに対応する。コンストレントの構造は以上にまたアペンディックスAに詳細に説明されている。

【0885】以上に詳細に説明したように、コンストレントは、(i) オブジェクトがメンバであるべきクラス、(ii) オブジェクトが特定されたクラスのインスタンスでもあるべきか否か、及び(iii) 特定されたクラスと関係なしにゼロがコンストレントを満たすか否か、を特定することによって、オブジェクトを制限する。あるコンストレントに対応するスタック5410の各々のオブジェクトは、そのオブジェクトがアーギュメントクラステストステップ5520中のコンストレントを満たすか否かを定めるために点検される。スタック5410のどれかのオブジェクトが定義オペレーションのプロパティ“arguments”の対応するコンストレントを満たさない場合には、処理はアーギュメントクラステストステップ5520から終了ステップ5520に移行し、そこでクラス“Argument Invalid”の例外が表明される。

【0886】逆に、スタック5410上の各々のオブジェクトが対応するコンストレントを満たす場合、処理はアーギュメントクラステストステップ5520からステップ5524に移行する。ステップ5524において、レスポndingオブジェクトのコンテキストにおいてサブジェクトオペレーションのインプリメンテーションを定義する方法オブジェクトが選定される。方法オブジェクトは、以下に一層詳細に説明するように、レスポndingオブジェクトがそのメンバであるクラスの中から選出される。

【0887】処理はステップ5524からステップ5526に移行し、そこで新しいユーザーによって定義されたフレーム5602(図116)が作り出される。レスポndingオブジェクト5604は、新しいユーザーによって定義されたフレーム5602のプロパティ“responder”とされる。ステップ5524において選択された方法オブジェクトのプロパティ“procedure”は、プロシージャ5606であり、新しいユーザーによって定義されたフレーム5602の、プロパティ“procedure”とされる。ユーザーによって定義されたフレーム5602のプロパティ“position”は、値が1に等しい整数5608となるようにイニシアライズされる。

【0888】プロパティ“stack”は、エンプティスタック5610にイニシアライズされる。

【0889】新しいユーザーによって定義されたフレー

ム5602が作り出されたら、処理はステップ5526からステップ5528に移行し、そこでオブジェクトは、ユーザーによって定義されたフレーム5304のスタック5410から、ユーザーによって定義されたフレーム5602のスタック5610に移動する。オブジェクトはスタック5410からスタック5610に移動し、その際に、オブジェクトの順序が保たれる。換言すれば、スタック5410上の最も上方のアーギュメントは、スタック5610の上部にあり、スタック5410の最も下方のアーギュメントは、スタック5610の底部にある。各々のオブジェクトは、検索された定義オペレーションのプロパティ“arguments”の対応するコンストレントのプロパティ“passage”に従って移動する。

【0890】プロパティ“passage”が“byRef”であれば、オブジェクトへのレファランスは、スタック5410からポップされ、スタック5610に保存される。したがって新しいユーザーによって定義されたプロシージャ5606は、リクエストしているユーザーによって定義されたフレーム5304のプロシージャ5406と同一の、アーギュメントへのアクセスを持つ。

【0891】プロパティ“passage”が“byProtectedRef”であれば、オブジェクトへのレファランスは、スタック5410からポップされ、保護されたレファランスとされ、スタック5610に保存される。したがって、新しいユーザーによって定義されたフレーム5602のプロシージャ5606は、アーギュメントを変更することができず、したがってアーギュメントは“リードオンリー”となる。

【0892】プロパティ“passage”が“byUnprotectedRef”であれば、リクエストするユーザーによって定義されたフレーム5304のスタック5410上のアーギュメントへのレファランスが保護されているか否かが定められる。このチェックは、アベンディックスAに一層詳細に記述されているアトリビュート“isProtected”を質疑することによって行う。レファランスが保護されている場合、クラス“レファランス保護”の例外が表明される。その他の場合、レファランスはスタック5410からポップされ、スタック5610に保存される。したがって新しいユーザーによって定義されたフレーム5602のプロシージャ5606には、保護されないレファランスによって追加されたアーギュメントを変更するためにアクセスが与えられる。

【0893】プロパティ“通過”は“バイコピー”であれば、スタック5410上に見出されるオブジェクトのコピーが作られ、そのコピーへの保護されないレファランスは、新しいユーザーによって定義されたフレーム5602のスタックに保存される。したがって、プロシージャ5606には、コピーによって通過させられるがアーギュメント自身にはアクセスを持たないアーギュメン

トのコピーへの無制限のアクセスが与えられる。

【0894】したがって、新しいユーザーによって定義されたフレーム5602は完全になり、サブジェクトオペレーションの遂行の直前のサブジェクトオペレーションの動的状態を表したものになる。処理はステップ5528（図55）からステップ5530に移行し、ここで、新しいユーザーによって定義されたフレーム5602は、スタック5302（図117）に保存されることによってプロセス5300の実行状態に含められる。したがって、スタック5410と、そのプロパティ“procedure”がプロシージャ5406である方法オブジェクトとは、それぞれ最早現在のスタックでも現在の方法でもなくなる。スタック5610は、現在のスタックであり、そのプロパティ“procedure”がプロシージャ5606である方法オブジェクトは、現在の方法である。処理はステップ5530（図55）からステップ5532に移行する。ステップ5532において、ユーザーによって定義されたフレーム5602のプロシージャ5606（図117）が遂行される。サブジェクトオペレーションの遂行とユーザーによって定義されたフレーム5602の生成とを生じさせる識別子がプロシージャ5406の遂行によって実行されるので、さらに別のオペレーションを生じさせる識別子が、ユーザーによって定義されたフレーム5602のプロシージャ5606の遂行によって引き起こされ、それによって、別のフレームが作り出され、これらの別のフレームは、後にスタック5302に保存され、そのプロシージャが後に遂行される。このように、プロセス5300の実行状態は、実行のスレッドを現すプロセス5300のプロパティ“フレーム”であるスタック5302上の1以上のフレームによって現される。

【0895】プロシージャ5606の遂行の後に、ユーザーによって定義されたフレーム5602は、サブジェクトオペレーションの遂行直後のサブジェクトオペレーションの動的状態を現している。

【0896】ユーザーによって定義されたフレーム5602のスタック5610は、サブジェクトオペレーションの遂行によって生じた結果（もしあれば）を有している。処理はステップ5532（図55）からステップ5534に移行し、ここでサブジェクトオペレーションを規定する定義オペレーションのプロパティ“result”が、定義オペレーションのアトリビュート“result”を質疑することによって生成される。処理はステップ5534から結果テストステップ5536に移行し、ここでクラス“コンストレント”のプロパティ“result”のメンバシップが定められる。換言すれば、プロパティ“result”は、コンストレントであってゼロでないことが検査される。プロパティ“result”がコンストレントであり、ゼロでない場合に、サブジェクトオペレーションは、結果を生じ、処理は結果テストステップ5536か

らスタックエンプティテストステップ5538に移行する。

【0897】スタックエンプティテストステップ5538において、エンジン解釈プロセス5300（図117）は、スタック5610にオブジェクトが存在するか否かを定める。スタック5610がエンプティであれば、処理はスタックエンプティテストステップ5538（図55）から終了ステップ5540に移行する。終了ステップ5540では、クラス“Result Missing”の例外が表明される。スタック5610（図117）がオブジェクトを含む場合、処理はスタックエンプティテストステップ5538（図55）から結果クラステストステップ5542に移行する。結果クラステストステップ5542において、エンジン解釈プロセス5300（図117）は、定義オペレーションのプロパティ“result”であるコンストレントをスタック5610上のオブジェクトが満たすか否かを定める。オブジェクトがコンストレントを満たさない場合、処理は結果クラステストステップ5542（図55）から終了ステップ5544に移行する。終了ステップ5544において、クラス“Result Invalid”の例外が表明される。オブジェクトがコンストレントを満たす場合、処理は結果クラステストステップ5542からステップ5546に移行する。

【0898】ステップ5546では、オブジェクトは、前述した定義オペレーションのプロパティ“passage”に従って、ユーザーによって定義されたフレーム5602のスタック5610（図117）からユーザーによって定義されたフレーム5304のスタック5410に移動する。

【0899】処理はステップ5546（図114）からステップ5548に移行する。

【0900】結果テストステップ5536においてプロパティ“result”がゼロである場合、処理は直接ステップ5548に移行する。ステップ5548では、新しいユーザーによって定義されたフレーム5602がスタック5302（図116）からポップされ、廃棄される（図115）。したがってスタック5410とそのプロパティ“procedure”がプロシージャ5406である方法オブジェクトとが、再びそれぞれ現在のスタックと現在の方法になる。処理はステップ5548（図114）からステップ5550に移行し、そこでプロシージャ5406の実行が、整数5408をインクリメントし、整数5408によって示された位置においてプロシージャ5406のアイテムを実行することによって継続される。したがって、あるオペレーションにレファランスする識別子の実行によって、そのオペレーションが遂行される。

【0901】定義オペレーション及び方法の選択  
前述したように、サブジェクトオペレーションと一緒に規定する定義オペレーションと方法オブジェクトとは、

それぞれステップ5510、5524において選択される（図113）。定義オペレーション及び方法の選択は、図97について前述した本発明の実施例のコンテキストにおいて説明される。この実施例においてはクラスオブジェクト4102は、クラス定義4100のプロパティ“Citation”、“interface”、及び“implementation”を複製することによって生成される。選択された定義オペレーションはインターフェースオブジェクトから選択され、選択された方法オブジェクトはインプリメンテーションオブジェクトから選択される。定義オペレーション及び方法オブジェクトは、各々フローチャート5700（図118）に従って選択される。適切な定義オペレーション及び方法オブジェクトは、レスポndingオブジェクトがその1つのインスタンスであるクラスに依存する。レスポndingオブジェクトは、それがクラスオブジェクトであるか否か、すなわちクラス“class”の1つのメンバであるか否かについて、クラステストステップ5702において検査される。

【0902】レスポndingオブジェクトはクラスオブジェクトであれば、処理はクラステストステップからクラスルックアップステップ5704に移行する。クラスルックアップステップ5704において、レスポndingオブジェクトのプロパティ“classFeatures”とレスポndingオブジェクトのインターフェーススーパークラスが以下に説明するようにサーチされる。クラスフィーチャのレキシコンすなわちプロパティ“classFeatures”はサブジェクトオペレーションの識別子すなわち実行識別子に適合するキーについてサーチされる。またクラスルックアップステップ5704において、レスポndingオブジェクトのプロパティ“classMembers”とレスポndingオブジェクトのインプリメンテーションスーパークラスが、実行識別子に関連した方法オブジェクトについて、以下に述べるようにサーチされる。

【0903】処理はクラスルックアップステップ5704からテストステップ5706に移行し、ここで、現在のプロセス例えばプロセス5300（図115）を解釈するエンジンは、定義オペレーション及び方法オブジェクトがクラスルックアップステップ5704において見出されるか否かを定める。サブジェクトオペレーションを定義する方法オブジェクト及び定義オペレーションが見出されたら、処理はテストステップ5706から終了ステップ5708に移行し、そこでセレクトioプロセスが終了する。その他の場合には、処理はテストステップ5706からインスタンスルックアップステップ5710に移行する。さらに、レスポndingオブジェクトがクラスオブジェクトでない場合、すなわちクラス“class”のメンバでない場合、処理はクラステストステップ5702から直接インスタンスルックアップステップ5710に移行する。

【0904】インスタンスルックアップステップ5710において、レスポンドオブジェクトが1つのインスタンスであるクラスのインターフェースオブジェクト及びそのクラスのインターフェーススーパークラス（複数）のインターフェースオブジェクト（複数）のプロパティ“インスタンスフィーチャーズ”が、実行識別子に関連したフィーチャ定義についてサーチされる。さらに、インスタンスルックアップステップ5710において、レスポンドオブジェクトがその1つのインスタンスであるクラスのインプリメンテーションと、そのクラスのインプリメンテーションスーパークラス（複数）のインプリメンテーション（複数）のプロパティ“インスタンスメソッズ”が、実行識別子に関連した方法オブジェクトについてサーチされる。スーパークラス（複数）が、サーチされる順序については以下に説明する。

【0905】レスポンドオブジェクトがクラスオブジェクトである場合には、クラス方法は、クラスフィーチャをインプリメントし得るに過ぎず、またインスタンス方法はインスタンスフィーチャをインプリメントし得るに過ぎない。

【0906】処理はインスタンスルックアップステップ5710から第2のテストステップ5712に移行し、そこでエンジンは定義オペレーションと方法オブジェクトとがインスタンスルックアップステップ5710に見出されるか否かを定める。実行識別子によって特定された定義オペレーションと方法オブジェクトが見出されたら、処理は第2のテストステップ5712から終了ステップ5716に移行する。終了ステップ5716において、選択プロセスは成功のうちに終了する。さもなければ、処理は第2のテストステップ5712から終了ステップ5714に移行する。終了ステップ5714においては、クラス“Feature Unavailable”の例外が表明される。

【0907】引用によって全体が本明細書の一部とされるアペンディックスAに開示されている予め定義されたハイアラキーの一部は、適切な定義オペレーション及び方法オブジェクトのために種々のクラスがサーチされる順序を示すために図119、図120に示されている。図119、図120は、予め定義されたインターフェースハイアラキーの一部を示すが、インプリメンテーションハイアラキーをサーチする方法は、インターフェースハイアラキーのサーチに関連して以下に説明される。

【0908】ハイアラキーグラフ5800（図119）はクラス“List”を現すクラスオブジェクト5802を示している。クラス“List”は、それぞれクラスオブジェクト5804、5806によって現されたミックスインスーパークラス“ordered”及びフレイバスーパークラス“コレクション”から引き継がれる。クラス“コレクション”はクラスオブジェクト5808によって現さ

れるスーパークラス“Object”から引き継がれる。

【0909】クラス“Object”は、クラスオブジェクト5810によって現されるミックスインスーパークラス“レファランسد”から引き継がれる。図119、図120において(i) 二重線は、フレイバクラスとフレイバクラスのサブクラスとの間に引かれ、(ii) 単一の実線は、ミックスインクラスとミックスインクラスのサブクラスとの間に引かれ、(iii) 斜線は、あるクラスとそのクラスのインスタンスとの間に引かれている。前述したように、インターフェースオブジェクト又はインプリメンテーションオブジェクトのプロパティ“superclasses”の最後のアイテムはミックスインクラスを特定すべきではない。この理由については以下に説明する。

【0910】例示の目的のために、フローチャート5700（図118）のコンテキストにおいてレスポンドオブジェクトが、クラス“List”を現すクラスオブジェクト5802（図119）であるものと想定する。クラスルックアップステップ5704（図118）において、クラスオブジェクト5802のプロパティ“interface”であるインターフェースオブジェクトのプロパティ“classFeatures”は、実行中の識別子をキーとする関連についてサーチされる。このような関連が見出されたら、識別子と関連されたフィーチャ定義はサブジェクトオペレーションのインターフェースを定義する定義オペレーションであり、選択され、定義オペレーションのサーチは終了する。

【0911】そのような関連が見出されなかったら、クラス“List”の第1のスーパークラスすなわちクラス“ordered”のインターフェースオブジェクトが、適切な定義オペレーションの場合と同様の仕方ですらサーチされる。適切に特定された定義オペレーションが見出されなかったら、インターフェースハイアラキーのデプスファーストウォークにおいてクラス“ordered”のスーパークラスがサーチされる。クラス“ordered”はスーパークラスを持たないので、クラス“List”のインターフェースオブジェクトすなわちクラス“コレクション”のプロパティ“superclasses”の次のアイテムが前述したようにサーチされる。サーチされる最後のスーパークラスはミックスインクラスではないので、フレイバスーパークラスのサーチの前に各々のミックスインスーパークラスをサーチする。

【0912】したがって、クラス“List”について前述したように、図119のクラスオブジェクトの残りの部分が次の順序すなわちクラス“Object”を現すクラスオブジェクト5808、クラス“レファランسد”を現すクラスオブジェクト5810、の順序でサーチされる。

【0913】サーチオペレーションのインプリメンテーションを規定する適切な方法オブジェクトのサーチは、2つの例外を除いて、適切な定義オペレーションのサーチについて前述した通りである。第1に、あるクラスの

インプリメンテーションオブジェクトのプロパティ”クラスメソッド”が、適切に特定された方法オブジェクトについてサーチされる。第2に、クラス”List”の、インターフェーススーパークラスではなく、インプリメンテーションスーパークラスが、適切に特定された方法オブジェクトについてサーチされる。

【0914】単一のフレバスーパークラスがサーチされる最後のスーパークラスであることの重要性は、次の仮説的な例を検討すれば明らかとなる。一例として、フレバ”コレクション”を現すクラスオブジェクト5806とクラス”コレクション”のスーパークラスを現すクラスオブジェクトとが、ミックスインクラス”ordered（順序あり）”を現すクラスオブジェクト5804のサーチの前にサーチされるものと想定する。さらに、クラス”Object”を現すクラスオブジェクト5808が、クラス”ordered”によって適合された1以上の特徴、すなわち、クラスオブジェクト5804によってその方法が提供された1以上の特徴を規定していると想定する。この仮説的な例においては、クラスオブジェクト5808が常にクラスオブジェクト5804の前にサーチされるので、クラスオブジェクト5808によって定義された特徴についての方法は、常にクラスオブジェクト5808に見出される。したがってクラスオブジェクト5804がクラスオブジェクト5808によって定義されたフィーチャに適合することは不可能であろう。したがってあるフレバのフレバスーパークラスは、常に定義オペレーション又は方法についてサーチされるフレバの最後のスーパークラスである。

【0915】引き続き、フローチャート5700（図118）のコンテキストにおいてレスポンドイングオブジェクトが、クラス”List”を現すクラスオブジェクト5802（図119、図120）である、この例示的な例において、ハイアラキーグラフ5850（図120）は、インスタンスルックアップステップ5710（図118）においてサーチされるクラスを示している。

【0916】定義オペレーションとクラスオブジェクト5802（図120）のインスタンスフィーチャに対する方法オブジェクトのサーチは、前述したクラスフィーチャの方法オブジェクト及び定義オペレーションに対するサーチと次の2つの点で異なっている。第1に、あるクラスのインターフェースオブジェクトのプロパティ”インスタンスフィーチャーズ”は、適切に特定された定義オペレーションについてサーチされ、あるクラスのインプリメンテーションオブジェクトのプロパティ”インスタンスメソッズ”は、適切に特定された方法オブジェクトについてサーチされる。第2に、レスポンドイングオブジェクトすなわちクラスオブジェクト5802がその1つのメンバであるクラス（複数）がサーチされる。

【0917】クラスをサーチする方法は、前述したものと同様でありここでは繰り返さない。適切な定義オペ

レーション及び方法オブジェクトが見出されるか、又は以下に述べる全てのクラスがサーチされるかするまで、以下のクラスが、以下の順序においてサーチされる。(i) クラスオブジェクト5802が1つのインスタンスであるクラス”class”を現すクラスオブジェクト5802。(ii) クラス”class”のミックスインスーパークラスである、クラス”インターチェンジド”を現すクラスオブジェクト5816。(iii) ミックスインクラス”インターチェンジド”のミックスインスーパークラス、したがってクラス”class”のミックスインスーパークラスであるクラス”変化しない”を現すクラスオブジェクト5818。(iv) クラス”class”のミックスインスーパークラスであるクラス”サイテッド”を現すクラスオブジェクト5814。(v) クラス”class”のフレバスーパークラスであるクラス”Object”を現すクラスオブジェクト5808。(vi) クラス”Object”のミックスインスーパークラスであるクラス”レファランسد”を現すクラスオブジェクト5810。

【0918】図119、図120のクラスのサーチにおいて、実行識別子に関連した定義オペレーション又は方法オブジェクトが見出されない場合には、レスポンドイングオブジェクトについてオペレーションが定義されず、クラス”Feature Undefined”の例外が表明される。

【0919】前述したように、レスポンドイングオブジェクトがクラスでない場合には、クラスルックアップステップ5704（図118）はスキップされる。一例として、レスポンドイングオブジェクトが、クラス”List”のインスタンスであるリスト5802Aであれば、ハイアラキーグラフ5800（図119）に示されたクラス（複数）が前述したようにインスタンスルックアップステップ5712（図118）においてサーチされる。

【0920】本発明の一実施例において、インターフェースハイアラキーとインプリメンテーションハイアラキーとは互いに独立である。換言すれば、あるクラスのインターフェーススーパークラスは、必ずしもそのクラスのインプリメンテーションスーパークラスではなく、あるクラスのインプリメンテーションスーパークラスは、必ずしもそのクラスのインターフェーススーパークラスではない。このかかる実施例において、オペレーション”makeClasses”の遂行の間にいかなるフィーチャ定義もインプリメンテーションなしにはならないようにすることをエンジンが確実にする。オペレーション”makeClasses”が失敗に終わり、クラス”classException”の例外が表明されるのは、次の条件の下である。すなわちこの条件とは、定義されるクラスが抽象的ではなく、定義されるクラスが、ある定義されたフィーチャをインプリメントする方法を定義せず、又はその方法をインプリメンテーションスーパークラスから受け継がないことである。すなわち、インターフェースハイアラキーにお

いて定義されたフィーチャはインプリメンテーションハイアラキーにおいて見出される方法によって常にインプリメントされる。

【0921】したがって、サブジェクトオペレーションのための適切な定義オペレーション及び方法オブジェクトは、前述したようにクラスハイアラキーをサーチすることによって見出されるか、又は例外が表明される。

#### 【0922】エスカレーション

前述したように、スーパークラスからあるフィーチャを受け継ぐクラスは、そのフィーチャに対する新しいインプリメンテーションを供与することによって、そのフィーチャのインプリメンテーションに代替される。しかしこの新しいインプリメンテーションは、スーパークラスによって提供されたインプリメンテーションを使用する必要はない。あるオブジェクトは、現在のクラスのイミディエイトスーパークラスの特別の1つ、又は任意の1つによってインプリメントされたフィーチャを、このフィーチャのエスカレートによって遂行するように指示される。

【0923】アペンディックスAに一層詳細に説明するように、現在のクラスは、現在実行されている方法を供与するクラスである。

【0924】あるフィーチャは、エスカレートされるべきフィーチャをレファランスする資格のある識別子を実行するか、又は“エスカレート”セレクトを実行することによってエスカレートされる。資格のある識別子は、フィーチャのインプリメンテーションを与えるべきクラスを特定するプロパティ“クオリファイアー”を含む。“エスカレート”セレクトの実行によってあるフィーチャがエスカレートされる場合、いかなるイミディエイトインプリメンテーションスーパークラスも、そのフィーチャのインプリメンテーションを供与しう。

【0925】なお、資格のある識別子のプロパティ“クオリファイアー”にアクセスを与えるいかなる属性も定義されていない。資格のある識別子が作り出され、その資格のある識別子が破壊されるまで変化しないままに保たれる場合に、資格のある識別子のプロパティ“クオリファイアー”が設定される。フィーチャのエスカレーションについてはアペンディックスAに詳細に説明されている。

【0926】エスカレーションはオブジェクトのイニシアライゼーションにおいて特に重要な役割をする。アペンディックスAは、本発明の各々の予め定義されたクラスのイニシアライゼーションの記述を含む。オブジェクトのイニシアライゼーションは、リストのイニシアライゼーションすなわちクラス“List”のインスタンスの例示的なコンテキストにおいて説明されるフローチャート5900（図121）に示されている。オペレーション“initialize”は、クラス“List”の各々のイミディエイトインプリメンテーションスーパークラスについて

エスカレートされる。前述したように、ハイアラキーダイアグラム5800（図119）は、クラス“List”のインターフェーススーパークラス（複数）を現している。さらにハイアラキーダイアグラム5800は、クラス“List”のインプリメンテーションスーパークラス（複数）の、正確に現している。

【0927】新しいリストがインスタンスであるクラスすなわちクラス“List”によって定義されたオペレーション“initialize”は、ステップ5902において遂行される。オペレーション“initialize”の遂行は、現在のスタックから0又は1以上のアーギュメントを消費する。処理はステップ5902からステップ5904に移行する。

【0928】ステップ5904においては、インプリメンテーションオブジェクトのアトリビュート“superClasses”を質疑することによって、クラス“List”のインプリメンテーションオブジェクトのプロパティ“superClasses”が生成される。前述したように、あるインプリメンテーションオブジェクトのプロパティ“superClasses”は、クラスをレファランスする識別子のリストである。処理はステップ5904からeach superclass ステップ5906に移行する。

【0929】each superclass ステップ5906及び次のステップ5910は、前記のようにして形成されたプロパティ“superClasses”の各々のスーパークラスがエスカレートステップ5908において処理されるループを形成する。すなわち、各々のスーパークラスにおいて、処理は、for each superclass ステップ5906からエスカレートステップ5908に移行する。エスカレートステップ5908においてスーパークラスは、フローチャート5900に従ってオペレーション“initialize”を遂行するように指示される。したがって、フローチャート5900は、インプリメンテーションハイアラキーのデプスファーストウオークにおいて反復的に遂行される。一例として、エスカレートステップ5908において“イニシアライズ”がエスカレートされた第1のイミディエイトスーパークラスは、(i) ステップ5902において“イニシアライズ”を実行し、(ii) ステップ5904においてスーパークラスのイミディエイトインプリメンテーションスーパークラス（複数）を生成させ、(iii) for each superclass ステップ5906及び次のステップ5910によって定義されたループにおいてスーパークラスの各々のスーパークラスに（イニシアライズ）をエスカレートする。

【0930】処理はエスカレートステップ5908から次のステップ5910を経てfor each superclass ステップ5906に移行する。したがってプロパティ“スーパークライシス”であるリストに見出される順序で、イニシアライゼーションが、各々のイミディエイトインプリメンテーションスーパークラスにエスカレートされ

る。クラス"List"のスーパークラスによるオペレーション"initialize"の各々の実行は、現在のスタックから0又は1以上のアーギュメントを消費し、スーパークラスのイミディエイトインプリメンテーションスーパークラスのイニシアライゼーションにエスカレートされる。

【0931】for each superclass ステップ5906及び次のステップ5910のループに従ってクラス"List"の各々のイミディエイトインプリメンテーションスーパークラスがオペレーション"initialize"を遂行した後は、処理はfor each superclass ステップ5906から終了ステップ5912に移行する。

【0932】終了ステップ5912において、クラス"List"によって定義されたオペレーション"initialize"の遂行は成功のうちに終了する。したがって、新しいリストが1つのメンバである各々のクラスは、前述したインスタンスルックアップステップ5710（図118）においてクラスがサーチされた順序で、オペレーション"initialize"を遂行する。

#### 【0933】モディファイアの実行

識別子の実行については以上に、そしてアペンディックスAに詳細に説明されている。しかしモディファイアが存在又は識別子の性質によって、識別子の実行が変更されることがある。例えば、アペンディックスAに述べたモディファイア代替ルールに従って、識別子がある属性を特定した場合、オペレーション"getAttribute"が遂行され、只1つのアーギュメントとして、実行中の識別子が消費される。種々のモディファイアと、識別子の実行に対する効果とについては、アペンディックスAに説明されており、この説明は引用によって本明細書の一部とされる。

#### 【0934】セレクトの実行

セレクトはその実行によってある特別のアクションが引き起こされるプリミティブである。各々のセレクトはアペンディックスAに説明されているが、幾つかのセレクトについては本明細書中において説明される。一例として、セレクト"break"、"continue"及び"succeed"は、プロシーダの遂行のコンテキストにおいて、以下に説明される。セレクト"self"の実行によって、現在のオブジェクトに対するレファレンス、すなわち現在の方法のレスポンド、が現在のスタックに保存される。図117のコンテキストにおいて、現在のオブジェクトは、フレーム5602のプロパティ"responder"であるオブジェクト5604であり、現在のスタックは、フレーム5602のプロパティ"stack"であるスタック5610であり、現在の方法は、そのプロパティ"procedure"がプロシーダ5606である方法オブジェクトである。

【0935】セレクト"Process"の実行によって、現在のプロセスに対するレファレンスが、現在のスタック

に保存される。図117において、現在のプロセスはプロセス5300である。現在のプロセスがエンジンプレイスであれば、0が現在のスタックに保存される。

【0936】したがって、本発明に従って作り出されたエンジンは、プロシーダのアイテムを順次実行することによって、プロシーダを実行する。プロシーダのアイテムの実行は、このアイテムが識別子である場合、その識別子によってレファレンスされるフィーチャを、前述した状態の元に行なわせる。遂行されるプロシーダの動的状態はフレームに記録される。ユーザーによって定義されるフィーチャをインプリメントする方法の動的状態は、ユーザーによって定義されたフレームに記録される。フィーチャのインターフェースとインプリメンテーションとをそれぞれ定義するフィーチャ定義及び方法オブジェクトは、レスポンドオブジェクトのメンバであるクラスから選択される。現在のクラスのスーパークラスによって提供されたインプリメンテーションは、フィーチャのエスカレートによって遂行される。一例として、オペレーション"initialize"は、反復的にエスカレートされ、レスポンドは1つのメンバである各々のクラスによって供与される方法が遂行される。モディファイアは、識別子の実行を変更するために用いられる。セレクトは、普通に行なわれる複雑なアクションが単一のオブジェクトの実行によって生ずるようになるための簡単な手段を提供する。これらのフィーチャは、ここに開示されたコンピューター命令セットにおいて高度の機能性及び一般性を組合せによって提供する。

#### 【0937】コントロールコンストラクト

前述したように、エージェントは、どのプレイス（1個以上）に移動し、そしてどのエージェント（1又は複数）と会合するかを定めるための複雑なロジックを適応することができる。ここに開示された命令セットは、その命令セットからなるプロシーダにおいて複雑で込み入ったロジックを実現することを可能にする高度の一般性を供与する複数の命令を供与する。

【0938】このように高度の一般性を与える以下のコントロールコンストラクトについて説明する。(i) リソース及びリソース割当フィーチャは、あるプロシーダの実行の間リソースの共有もしくは排他的コントロールを取得し保持する能力をプロセスに与える。(ii) デシジョンハンドリングフィーチャは、プロセスがプロシーダを条件的に実行することを可能にする。(iii) 例外ハンドリングフィーチャは、プロセスが例外の表明を検出してそれに対してアクションをとることを可能にする。(iv) ルーピングフィーチャはプロシーダを反復遂行してそれにより反復的なタスクを効率的に行なうことを可能にする。

#### 【0939】リソース及びリソース割当フィーチャ

以上に説明しアペンディックスAに説明するように、1以上のプロセスを処理するエンジンはそれを同時に行な



ている。一例として、エンジン132B（図50）は、エージェント150A及び150Bを同時に処理する。エンジン132Bは、エンジン132Bによって実行されるエージェント150Aによる第1のプロシーダの遂行の間のどの点でも、エージェント150Aによる第1のプロシーダの遂行を一時的に中止しておくことができる。エンジン132Bは、その場合、エージェント150Bのために第2のプロシーダを自由に実行することができる。エンジン132Bは、第2のプロシーダの遂行のどの点においても、エージェント150Bのために第2のプロシーダの実行を一時的に中断し、エージェント150Aのために第1のプロシーダの実行を再開することができる。このようにしてエンジン132Bは、エージェント150A、150Bを同時に処理する。

【0940】アペンディックスAに説明するように、あるエンジンは、予め規定されたフィーチャの大部分をアトミックに実行する。エンジンが別のプロセスを処理する目的のために、そのフィーチャが成功のうちに終了しまたそのフィーチャが成功のうちに終了するか、又は失敗するまで、そのフィーチャの実行を中止しておくことを拒んだ場合、そのフィーチャはアトミックに遂行される。あるフィーチャがアトミックに遂行される場合、そのフィーチャの遂行の間いかなる他のプロセスも解釈されないで、現在のプロセスを除くいかなるプロセスもリソースを変更したり操作したりすることはできない。

【0941】アトミックに遂行されないあらかじめ提示されたフィーチャは、アペンディックスAに説明されている。

【0942】ユーザーによって定義されたフィーチャをアトミックに遂行するメカニズムは存在しない。すなわち、ユーザーによって定義されたフィーチャ又はアトミックに遂行されないあらかじめ定義されたフィーチャの遂行の間にあるプロセスがリソースの共有又は排他的な使用を取得し得るようにするためのメカニズムが必要である。ユーザーは、クラス“リソース”によって定義されアペンディックスAに詳細に説明されているオペレーション“use”の使用によってリソースの共有又は排他的な使用が得られるようにプロセスを構成することができる。オペレーション“use”の遂行の動的状態はクラス“ユーザフレーム”のユーザフレームによって記録される。ユーザフレームは付録Bに記載されている。

【0943】実行されるオブジェクトの条件的な実行  
実行されるオブジェクトは、オペレーション“if”又はオペレーション“either”の遂行によって条件的に実行される。応答する実行されるオブジェクトによるオペレーション“if”の遂行は、ブーリアンオブジェクトを消費し、ブーリアンオブジェクトの値が“true”であれば、実行されるオブジェクトを遂行する。なを、プロシーダは、実行されるオブジェクト、すなわちミックス

インクラス“エグゼキューティッド”のメンバであり、プロシーダの遂行については図109、図110を参照して以上に説明したとおりである。

【0944】フローチャート6000（図122）は、ミックスインクラス“エグゼキューティッド”によって規定されたオペレーション“if”のインプリメンテーションを示している。オペレーション“if”を遂行する際に、応答する実行されるオブジェクトは、ステップ6002において、現在のスタックからブーリアンオブジェクトをポップする。処理はステップ6002からブーリアンテストステップ6004に移行し、ここで、ステップ6002でポップされたブーリアンオブジェクトが、“true”と比較される。ブーリアンオブジェクトが“true”であれば、処理は、ブーリアンテストステップ6004から6006に移行し、ここで応答する実行されるオブジェクトが遂行される。処理はステップ6006から、例外テストステップ6010に移行し、ここで、オペレーション“if”を実行しているエンジンが、応答する実行されるオブジェクトをステップ6006で実行したことによって例外が表明されたか否かを定める。

【0945】ステップ6006において例外が表明された場合、処理は例外テストステップ6010から終了ステップ6008に移行する。更に、ブーリアンオブジェクトが“false”である場合、処理はブーリアンテストステップ6004から直接に終了ステップ6008に移行する。終了ステップ6008ではオペレーション“if”が成功のうちに終了する。ステップ6006において例外が表明された場合、処理は例外テストステップ6010から終了ステップ6012に移行し、ここで、応答する実行されるオブジェクトがステップ6006において遂行されたことによって表明された例外が表明され、それによってオペレーション“if”が失敗となる。オペレーション“if”の遂行の動的状態は、あらかじめ規定されたフレームすなわち以上に説明したアペンディックスBにおいて説明されるクラス“あらかじめ定義したフレーム”の1つのメンバ、によって記録される。2つの実行されるオブジェクトのうちの1つが選択され、オペレーション“either”の使用によって実行される。実行されるオブジェクトは、ブーリアンオブジェクトの値が“true”であれば、第2の実行されるオブジェクトとブーリアンオブジェクトとを消費し、応答する実行されるオブジェクトの実行を生じさせ、ブーリアンオブジェクトの値が“false”であれば第2の実行されるオブジェクトの実行を生じさせることによって、オペレーション“either”を遂行する。フローチャート6100（図123）は、ミックスインクラス“エグゼキューティッド”によって規定されたオペレーション“either”のインプリメンテーションを示している。

【0946】応答する実行されるオブジェクトは、ステ

ップ6 1 0 2において第2の実行されるオブジェクトを現在のスタックからポップする。処理は、ステップ6 1 0 2からステップ6 1 0 4に移行し、ここで応答する実行されるオブジェクトは現在のスタックからブーリアンオブジェクトをポップする。処理はステップ6 1 0 4からブーリアンテストステップ6 1 0 6に移行し、ここで、ステップ6 1 0 4においてポップされたブーリアンが“true”と比較される。ブーリアンオブジェクトが“true”であれば処理はブーリアンテストステップ6 1 0 6からステップ6 1 0 8に移行し、ここで応答する実行されるオブジェクトが遂行される。

【0947】その逆に、応答するブーリアンオブジェクトの値が“false”であれば、処理はブーリアンステップ6 1 0 6からステップ6 1 1 0に移行し、ここで第2の実行されるオブジェクトが遂行される。処理はステップ6 1 0 8又はステップ6 1 1 0から終了ステップ6 1 1 2に移行する。終了ステップ6 1 1 2では、オペレーション“either”が終了し、ステップ6 1 0 8又はステップ6 1 1 0において表明された例外（もしあれば）が表明される。オペレーション“either”を遂行する実行されるオブジェクトの動的状態は、あらかじめ定義されたフレームによって記録される。

【0948】1以上の実行されるオブジェクトから選択され、オペレーション“select”の遂行によって遂行される。オペレーション“select”は、クラス“Object”によって定義され、遂行されるべき特別な実行オブジェクトを決定するオブジェクトによって遂行される。図1 2 5、図1 2 6は、オペレーション“select”のインターフェースを示している。図1 2 5は、オブジェクト6 2 0 4によって遂行されるオペレーション“select”の動的状態を記録するあらかじめ定義されたフレーム6 2 0 0の状態を示している。あらかじめ定義されたフレーム6 2 0 0が、プロセス（図示しない）の実行状態の一部であり、オペレーション“select”の遂行直前において示されている。

【0949】オブジェクト6 2 0 4はレスポндаである。あらかじめ定義されたフレーム6 2 0 0は、オブジェクト6 2 0 4を特定するプロパティ“responder”を持たない。その代わりに、オブジェクト6 2 0 4は、現在のプロセスのプロパティ“フレームズ”であるスタック（図示しない）上の現在のフレームの直下のフレーム（図示しない）のプロパティ“procedure”内のその位置によって、レスポндаとして特定される。オブジェクト6 2 0 4は、その位置が、そのフレームのプロパティ“position”よりも1だけ少ないプロシージャ（図示しない）のアイテムである。一例として、ユーザーによって定義されたフレーム5 6 0 2（図1 1 7）のレスポндаは、その位置が整数5 4 0 8の値よりも1だけ少ないプロシージャ5 4 0 6のアイテムである。

【0950】スタック6 2 0 2（図1 2 5）は現在のス

タックである。スタック6 2 0 2は、オペレーションのアーギュメント、及び1以上の対のオブジェクト及びそれに関連した実行されるオブジェクトのアーギュメントを表現する、文字“M”によって位置が示されたマークである。一例としてスタック6 2 0 2は、上部からマークの直上のところにかけて、オブジェクト6 2 0 6と実行されるオブジェクト6 2 0 8とオブジェクト6 2 1 0と実行されるオブジェクト6 2 1 2とゼロ6 2 1 4と実行されるオブジェクト6 2 1 6とを収容している。オブジェクト6 2 0 6、6 2 1 0及びゼロ6 2 1 4はそれぞれの実行されるオブジェクト6 2 0 8、6 2 1 2及び6 2 1 6に関連されている。

【0951】オペレーション“select”のインプリメンテーションは、フローチャート6 3 0 0（図1 2 4）によって示される。ステップ6 3 0 2において、オブジェクト6 2 0 6ー7 6 1 6（図1 2 5）及びマークはスタック6 2 0 2からポップされる。処理はステップ6 3 0 2からステップ6 3 0 4に移行し、ここで、オブジェクト6 2 0 6、6 2 1 0及びゼロ6 2 1 4をキーとし、それぞれの実行されるオブジェクト6 2 0 8、6 2 1 2及び6 2 1 6を関連された値とするディクショナリーが形成される。処理はステップ6 3 0 4からステップ6 3 0 6に移行する。ステップ6 3 0 6において、その関連されたキーがレスポディングオブジェクト6 2 0 4に等しい実行されるオブジェクトが、ステップ6 3 0 4において形成されたディクショナリーから検索される。ディクショナリーのキーの与えられた場合のディクショナリーの値の検索についてはアペンディクスAに一層詳細に説明されている。

【0952】処理はステップ6 3 0 6からテストステップ6 3 0 8に移行する。テストステップ6 3 0 8ではオペレーション“select”を実行しているエンジンが、実行されるオブジェクトがディクショナリーから成功のうちに検索されているか否かを定める。応答するオブジェクト6 2 0 4に等しいキーに組み合わせられた実行されるオブジェクトがディクショナリー内において見いだされ検索された場合、処理はステップ6 3 1 4に移行する。ステップ6 3 1 4では検索された実行されるオブジェクトが遂行される。処理はステップ6 3 1 4から例外テストステップ6 3 1 6に移行しここでエンジンは、実行されるオブジェクトの遂行がステップ6 3 1 4において成功のうちに完了したか否かを定める。

【0953】ステップ6 3 1 4において実行されるオブジェクトの遂行が失敗し、例外が表明された場合、処理は例外テストステップから終了ステップ6 3 2 0に移行する。終了ステップ6 3 2 0では例外が表明され、オペレーション“select”は失敗となる。しかし検索された実行されるオブジェクトがステップ6 3 1 4において成功のうちに遂行された場合、処理は例外テストステップ6 3 1 8から終了ステップ6 3 1 6に移行する。

【0954】終了ステップ6316ではオペレーション”select”は成功のうちに終了する。

【0955】テストステップ6308では、レスポンドオブジェクト6204に等しいキーが、ステップ6304において形成されたディクショナリーに見いだされなかった場合、処理はテストステップ6308からステップ6310に移行する。ステップ6310では、関連するキーがゼロであるディクショナリー内に含まれる実行されるオブジェクト、例えばゼロ6214に組み合わせられたオブジェクト6216、が検索される。処理はステップ6310から第2のテストステップ6312に移行し、ここでエンジンは、実行されるオブジェクトがステップ6310において成功のうちに検索されたか否かを定める。ゼロに等しい組み合わせられたキーを有するディクショナリー内の実行されるオブジェクトが見いだされ、成功のうちに検索されたら、処理は第2のテストステップ6312からステップ6314に移行しここで前述のように実行されるオブジェクトが遂行される。

【0956】ステップ6310において実行されるオブジェクトが検索されなかった場合、オブジェクト6204に等しいキー及びゼロに等しいキーがディクショナリーにおいて見いだされなかったので、処理は第2のテストステップ6312から終了ステップ6316に移行する。前述したように、終了ステップ6316では、オペレーション”select”は成功のうちに終了する。すなわちレスポンドオブジェクトに等しいキーに組み合わせられた実行されるオブジェクトがなく、又はゼロに組み合わせられている場合、オペレーション”select”は、ディクショナリー内のいかなる実行されるオブジェクトも遂行することなく、成功のうちに終了する。

【0957】図126は、レスポンドオブジェクト6204によるオペレーション”select”の遂行の直後においてのセレクトフレーム6200の状態を示している。オペレーション”select”の遂行は、図126のエンピティスタック6202に示すようにいかなる結果も生じていない。

【0958】実行されるオブジェクトは、オペレーション”while”の遂行によって条件が満たされるまで繰り返し実行される。オペレーション”while”の遂行の動的状態は、あらかじめ定義されたフレーム6402（図127）に記録される。実行されるオブジェクト6404は、オペレーション”while”のレスポンドであり、応答する実行されるオブジェクトである。

【0959】実行されるオブジェクト6404は、あらかじめ定義されたフレーム6402のプロパティ”procedure”である。実行されるオブジェクト6406は、あらかじめ定義されたフレーム6402のプロパティ”precondition”であり、いかに説明するようにアーギュメントとして消費される実行されるオブジェクトである。

【0960】オペレーション”while”は、アーギュメントとして実行されるオブジェクト6406を消費する、応答する実行されるオブジェクト6404によって遂行される。オペレーション”while”の遂行は、フローチャート6500（図128）によって示されている。ステップ6502において、実行されるオブジェクト6406は現在のスタックからポップされる。処理はステップ6502からステップ6504に移行し、ここで実行されるオブジェクト6406が遂行される。実行されるオブジェクト6406がステップ6504で遂行されることによって、応答する実行されるオブジェクト6404が遂行されるか否かを示すブーリアンが現在のスタックに保存される。

【0961】処理はステップ6504から例外テストステップ6506に移行し、ここで、オペレーション”while”を遂行するエンジンが、実行されるオブジェクト6406の遂行によって例外が表明されるか否かを定める。実行されるオブジェクト6406の遂行が例外を表明する場合、処理は例外テストステップ6506から終了ステップ6508に移行し、ここで、オペレーション”while”によって例外が表明され、オペレーション”while”は失敗となる。しかし、実行されるオブジェクト6406が成功のうちに遂行された場合、処理は例外テストステップ6506からステップ6510に移行する。

【0962】ステップ6510では、ブーリアンが現在のスタックからポップされる。処理はステップ6510から真テストステップ6512に移行する。真テストステップ6512では、現在のスタックからポップされたブーリアンが”true”と比較される。もしもブーリアンが”true”であれば、処理は真テストステップ6512からステップ6514に移行し、ここで応答する実行されるオブジェクト6406が遂行される。処理はステップ6514から第2の例外テストステップ6516に移行し、ここでエンジンは、実行されるオブジェクト6406が例外を表明するか否かを定める。

【0963】実行されるオブジェクト6406の遂行によってステップ6514で例外が表明されると、処理は第2のテストステップ6516から継続テストステップ6522に移行する。継続テストステップ6522では、表明された例外は、以下にまたアペンディクスAに説明されるセレクト”継続”の実行によって表明された内部的な例外”継続”と比較される。以下に一層詳細に説明するように、セレクト”継続”の実行は、1つのプロセスの反復的な遂行を終了させ、そのプロセスの新しい反復的な遂行を開始させる。ステップ6514において表明された例外が、内部的な例外”継続”であれば、処理は継続テストステップ6522から、前述したステップ6504に移行する。

【0964】その反対に、ステップ6514において表

明された例外が内部的な例外”継続”でなければ、処理は継続テストステップ6522からブレイクテストステップ6524に移行する。ブレイクテストステップ6524では、ステップ6514において表明された例外が、以下にそしてアペンディクスAに説明されるセレクト”ブレイク”の実行によって表明された内部的な例外”ブレイク”と比較される。以下に一層詳細に説明するように、セレクト”ブレイク”の実行によって、プロセスの反復的な遂行が、そのプロセスの新しい反復的な遂行を開始することなしに終了される。ステップ6514において表明された例外が内部的な例外”ブレイク”であれば、処理は継続テストステップ6522から終了ステップ6520に移行し、ここでオペレーション”while”は、以下に説明するように成功のうちに終了する。

【0965】その逆に、ステップ6514において表明された例外が内部的な例外”ブレイク”でない場合、処理はブレイクテストステップ6524から終了ステップ6518に移行する。終了ステップ6518では、ステップ6514において表明された例外が表明され、オペレーション”while”を失敗に終わらせる。

【0966】しかし、実行されるオブジェクト6404の遂行が成功し、ステップ6514において例外が表明されなかった場合には、処理は第2の例外テストステップ6516から、ステップ6504に移行し、そこで実行されるオブジェクト6406は再び遂行される。

【0967】ステップ6504、6506、6510、6512、6514、6516、6522の遂行は、実行されるオブジェクト6406の遂行が”false”の値を有するブーリアンを生じるか、又はステップ6508又はステップ6518において例外が表明されるまで反復される。実行されるオブジェクト6406の遂行が”false”であるブーリアンを生じた場合には、処理は真テストステップ6512から終了ステップ6520に移行し、そこでオペレーション”while”は成功のうちに終了する。従ってプロセス6406は、実行されるオブジェクトの遂行が”true”のブーリアンを生じている間は反復して遂行される。

【0968】このように、オペレーション”if”、”either”、”select”、及び”while”が、ここに開示されたコンピュータ命令のセットに、実質的な決定能力を供与する。

【0969】例外ハンドリングフィーチャ  
アペンディクスAにおいて詳細に説明するように、オペレーション”do”を遂行する実行されるオブジェクトは、それによって遂行され、この実行されるオブジェクトの遂行が例外を表明した場合、オペレーション”do”によって例外が表明され、オペレーション”do”が失敗となる。例外の伝搬は、前述したようにフローチャート5200（図109）のステップ5210、5212に

示されている。実行されるオブジェクトの遂行の失敗は、一般に、実行を直接に、又は間接に引き起こす実行されるオブジェクトの失敗を生じさせる。あるプロセスのオペレーション”live”を失敗させる例外は、プロセスの破壊を生ずる。従って例外の伝搬を検出してこれを防止するための命令を用意することが好ましい。

【0970】第2の実行されるオブジェクトの遂行を引き起こす、第1の実行されるオブジェクト、例えばプロセスは、第2の実行されるオブジェクトによって表明される例外を”キャッチ”することによって、第2の実行されるオブジェクトによって表明された例外による第1の実行されるオブジェクトの失敗を防止することができる。

【0971】例外を”キャッチ”することは、例外の表明を検出し、このような事態において特定のアクションがとられることを指示することである。第2の実行されるオブジェクトによって表明され、第1の実行されるオブジェクトによってキャッチされる例外は、第1の実行されるオブジェクトを失敗に終わらせない。

【0972】実行されるオブジェクトの遂行によって表明される例外は、オペレーション”do”の場所でオペレーション”catch”の使用によって実行されるオブジェクトの遂行を生じさせることによってcatchされる。オペレーション”catch”によって遂行が生ずる、実行されるオブジェクトは、実行されるオブジェクトがオペレーション”do”を遂行しているかのように遂行されるが、実行されるオブジェクトの遂行によって表明された例外がオペレーション”catch”を失敗に終わらせることはない。この場合には、現在のスタックに例外が保存され、結果として返却される。

【0973】オペレーション”catch”のインターフェースは、図129、図130によって示される。図129は、キャッチフレーム6602の状態を示し、このキャッチフレーム6602は、実行されるオブジェクト6604によるオペレーション”catch”の遂行の直前にミックスインクラス”エグゼキューティッド”によって定義されたオペレーション”catch”の動的状態を記録する。キャッチフレーム6602は、あるプロセス（図示しない）の実行状態の一部である。実行されるオブジェクト6604は、応答する実行されるオブジェクトであり、キャッチフレーム6602のプロパティ”procedure”である。オペレーション”catch”が遂行されると、スタック6606の頂点にあるクラスオブジェクト6608である単一のアーギュメントが消費される。スタック6606は現在のスタックである。スタック6606は、現在のプロセスのプロパティ”フレームズ”であるスタック（図示しない）のキャッチフレーム6602の下方にあるもっとも上方のユーザーによって定義されるフレーム（図示しない）のプロパティ”stack”である。クラスオブジェクト6608は、クラス”例外”

又はそのサブクラスを表す。

【0974】オペレーション”catch”のインプリメンテーションは、フローチャート6700（図131）によって示される。クラスオブジェクト6608は、スタック6606からポップされることによって、ステップ6702において消費される。

【0975】処理はステップ6702からステップ6704に移行し、そこで実行されるオブジェクト6604は遂行される。処理はステップ6704から例外テストステップ6706に移行し、ここで、オペレーション”catch”の遂行を行うエンジンは、実行されるオブジェクト6604の遂行が例外を表明するか否かを定める。実行されるオブジェクト6604の遂行が例外を表明しなかった場合、処理は例外テストステップ6706からステップ6708に移行する。ステップ6708では、スタック6606にゼロが保存される。処理はステップ6708から終了ステップ6710に移行し、そこでオペレーション”catch”を、成功のうちに終了する。

【0976】実行されるオブジェクト6604をステップ6704で実行することが失敗し、例外が表明されたら、処理は例外テストステップ6706からメンバテストステップ6712に移行する。メンバテストステップ6712では、エンジンは、クラスオブジェクト6608によって表されたクラス中のステップ6704において表明される例外のメンバシップをチェックする。表明された例外がクラスオブジェクト6608によって表されるクラスのメンバでない場合、処理はメンバテストステップ6712から終了ステップ6714に移行する。終了ステップ6714では、ステップ6706において表明された例外が、オペレーション”catch”によって表明され、オペレーション”catch”は失敗となる。表明された例外がクラスオブジェクト6608によって表されたクラスのメンバである場合、処理はメンバテストステップ6712からステップ6716に移行する。ステップ6716では、ステップ6704で表明された例外がスタック6606に保存され、処理はステップ6716から終了ステップ6710に移行する。前述したように、終了ステップ6710では、オペレーション”catch”は成功のうちに終了する。従って表明された例外はキャッチされ、オペレーション”catch”は成功し、表明された例外が伝搬されることはない。

【0977】図130は、プロシージャ6604によるオペレーション”catch”の遂行直後のキャッチフレーム6602の状態を示している。

【0978】スタック6606の上部には、例外6610があり、この例外6610が、プロシージャ6604の遂行が失敗した場合にプロシージャ6604の遂行によって表明される例外である。この場合、例外6610は、クラスオブジェクト6608（図129）によって表されるクラスのメンバである。プロシージャ6604

の遂行が成功した場合に、スタック6606の例外6610（図130）の場所にあるのはゼロ（図示しない）である。従ってここに開示されたコンピュータ命令セットは、例外を検出し取り扱うための手段を備えている。

#### 【0979】ルーピングフィーチャ

ここに開示されたコンピュータ命令セットは、実行されるオブジェクト例えばプロシージャを繰り返し遂行して、反復タスクの遂行を容易にするための手段を備えている。前述したオペレーション”while”は、ルーピングフィーチャの一例である。そのほかに、実行されるオブジェクトは、オペレーション”loop”及び”repeat”を用いて繰り返し遂行される。

【0980】オペレーション”loop”は、応答する実行されるオブジェクトを繰り返し遂行することによって、実行されるオブジェクトを不特定に遂行する。実行されるオブジェクトによるオペレーション”loop”の遂行は、アーギュメントを消費せず、いかなる結果も生じない。実行されるオブジェクトによるオペレーション”repeat”の遂行は、整数アーギュメントを消費し、消費されるアーギュメントの整数の値に等しい回数だけ、応答する実行されるオブジェクトを実行し、いかなる結果も生じない。オペレーション”loop”又はオペレーション”repeat”の間に応答する実行されるオブジェクトが遂行されることはここでは実行されるオブジェクトの”反復的な遂行”と呼ばれる。

【0981】フローチャート6800（図132）は、応答する実行されるオブジェクトによって遂行されるオペレーション”loop”のインプリメンテーションを示している。応答する実行されるオブジェクトはステップ6802において遂行される。処理はステップ6802から例外テストステップ6804に移行し、そこで、オペレーション”loop”を遂行するエンジンは、応答する実行されるオブジェクトが例外を表明するか否かを定める。

【0982】例外が表明されなかった場合すなわち応答する実行されるオブジェクトが成功のうちに遂行された場合、処理は例外テストステップ6804からステップ6802に移行し、そこで応答する実行されるオブジェクトが再び遂行される。しかし、応答する実行されるオブジェクトの遂行が例外を表明した場合には、処理は例外テストステップ6804から継続テストステップ6806に移行する。継続テストステップ6806において、ステップ6802で応答する実行されるオブジェクトの遂行によって表明された例外は、内部的な例外”継続”と比較される。内部的な例外”継続”及び”ブレーク”は、以下に及びアペンディクスAに一層詳細に説明されるセレクト”継続”及び”ブレーク”の実行によってそれぞれ表明される。これらの例外は、内部的な例外”継続”及び”ブレーク”がそれに応答してアクションを行うエンジンによって検出されるので、”内部的”

と呼ばれる。第3の内部的な例外すなわちセクタ”成功”の実行の結果として表明される内部的な例外”成功”については以下に説明する。内部的な例外は、オペレーション”catch”によってはキャッチされず、オペレーション”live”を失敗に終わらせることによって以下に説明する場合を除いてプロセスの破壊を生じさせる。

【0983】ステップ6802において、実行されるオブジェクトの遂行によって表明された例外が内部的な例外”継続”であると、処理は継続テストステップ6806からステップ6802に移行し、そこで応答する実行されるオブジェクトが再び遂行される。そうでなく、表明された例外が内部的な例外”継続”でなければ、処理は継続テストステップ6806からブレイクテストステップ6808に移行する。ブレイクテストステップ6808では、応答する実行されるオブジェクトを実施することによって表明された例外は、内部的な例外”ブレイク”と比較される。ステップ6802において、実行されるオブジェクトの遂行によって表明された例外が内部的な例外”ブレイク”であると、処理はブレイクテストステップ6808から終了ステップ6810に移行し、そこでオペレーション”loop”は成功のうちに終了する。そうでなくて、表明された例外が内部的な例外”ブレイク”でない場合には、処理はブレイクテストステップ6808から終了ステップ6812に移行し、そこでオペレーション”loop”は失敗に終わり、ステップ6802において応答する実行されるオブジェクトの遂行によって表明された例外は、オペレーション”loop”によって表明される。

【0984】図133は、実行されるオブジェクト6904によるオペレーション”repeat”の遂行の動的状態を記録するリピートフレーム6902を示している。リピートフレーム6902は、あるプロセス（図示しない）の実行状態の一部分である。リピートフレーム6902は、プロパティ”procedure”、”repetitions”、”repetitionsSoFar”及び”position”を含む。実行されるオブジェクト6904は、オペレーション”repeat”のレスポンドであり、リピートフレーム6902のプロパティ”procedure”である。整数6910、6911は、リピートフレーム6902のそれぞれプロパティ”repetitionsSoFar”及び”repetitions”である。整数6910は、実行されるオブジェクト6906の終了した反復遂行の回数を示し、整数6911は、消費された整数のアーギュメントによって特定される反復遂行の全回数である。

【0985】フローチャート7000（図134）は、実行されるオブジェクト6904によって遂行されるオペレーション”repeat”のインプリメンテーションを示している。ステップ7002において、整数6910はゼロにイニシャライズされ、整数6911は、アーギュ

メントとして消費された整数の値にイニシャライズされる。処理はステップ7002からテストステップ7004に移行し、そこで整数6910は整数6911と比較される。整数6910が整数6911に等しいかこれよりも大きければ、処理はテストステップ7004から終了ステップ7018に移行し、そこでオペレーション”repeat”は成功のうちに終了する。従って、消費される整数が整数でなければ、オペレーション”repeat”の遂行は効果を持たない。そうでないと、すなわち整数6910が整数6911より小さいと、処理はテストステップ7004からステップ7005に移行し、そこで整数6910はインクリメントされる。処理はステップ7005からステップ7006に移行し、そこで整数6910に等しい値を持つ整数オブジェクトが現在のスタックに保存される。

【0986】処理はステップ7006からステップ7008に移行し、そこで応答する実行されるオブジェクト6904が遂行される。現在のスタックから整数をポップすることは、実行されるオブジェクト6904の仕事である。換言すれば、実行されるオブジェクト6904の遂行が現在のスタックから整数をポップしない場合には、整数は、オペレーション”repeat”の遂行の間現在のスタックに留まっている。

【0987】処理はステップ7008から例外テストステップ7010に移行し、そこで、オペレーション”repeat”の遂行を行うエンジンは、応答する実行されるオブジェクト6904の遂行が例外を表明するか否かを定める。例外が表明されなかったら、処理は例外テストステップ7010からテストステップ7004に移行する。その逆に、例外が表明された場合、処理は例外テストステップ7010から、継続テストステップ7014に移行し、そこで例外が内部的な例外”継続”と比較される。表明された例外が内部的な例外”継続”であれば、処理は継続テストステップ7014からテストステップ7004に移行する。このように、ステップ7008において例外が表明されなかったり、表明された例外が内部的な例外”継続”である場合には、処理はテストステップ7004に移行し、そこで整数6910が整数6911と再び比較される。従って、セクタ”継続”の実行は、早すぎる時期に、応答する実行されるオブジェクト6904の反復遂行を終了させ、オペレーション”repeat”の遂行においてその後の反復的な遂行に影響することはない。

【0988】継続テストステップ7014において、ステップ7008で応答する実行されるオブジェクト6904の遂行が内部的な例外”継続”以外の例外を表明したことをエンジンが定めた場合、処理は継続テストステップ7014からブレイクテストステップ7016に移行する。ブレイクテストステップ7016において、表明された例外が内部的な例外”ブレイク”と比較され

る。表明された例外が内部的な例外”ブレーク”であれば、処理はブレークテストステップ7016から終了テストステップ7018に移行する。

【0989】終了テストステップ7018において、オペレーション”repeat”は成功のうちに終了する。従って、セレクト”ブレーク”の実行は、応答する実行されるオブジェクト6904の反復的な遂行を終了させ、オペレーション”repeat”の遂行を成功のうちに終了させ、オペレーション”repeat”の遂行においてその後の反復的な遂行を全てアボートする。

【0990】表明された例外が、内部的な例外”ブレーク”でない場合には、処理はブレークテストステップ7016から終了ステップ7020に移行する。

【0991】ステップ7020では、応答する実行されるオブジェクトの6904の遂行によって表明された例外は、オペレーション”repeat”によって表明され、オペレーション”repeat”を失敗させ、例外を伝搬させる。前述したように、オペレーション”catch”は、例外の伝搬を防止するために使用することができる。整数6910がステップ7005において、ゼロに等しいか、又はゼロより小さい値までデクリメントされると、処理はテストステップ7004から終了ステップ7018に移行し、そこでオペレーション”repeat”が成功のうちに終了する。実行されるオブジェクトによるオペレーション”repeat”の遂行は、図134においてステップ7008によって実行されるオブジェクトの遂行から結果したもの以外の結果を生じさせない。

【0992】セレクト”継続”の実行によって、内部的な例外”継続”が表明され、この内部的な例外は、オペレーション”while”、例えば継続テストステップ6522（図128）、オペレーション”loop”、例えば継続テストステップ6806（図132）、及びオペレーション”repeat”例えば継続テストステップ7014

（図134）によってキャッチされる。セレクト”継続”の実行による内部的な例外”継続”の表明は、オペレーション”while”、”loop”、又は”repeat”を遂行している実行されるオブジェクトの反復的な遂行を終了させる。先により詳細に述べたように、オペレーション”while”、”loop”又は”repeat”を遂行している間に、内部的な例外”継続”の検出によって、応答するプロシーダのその後の反復的な遂行例えば図128、図132及び図134の継続テストステップ6522、606、7014のような反復的な遂行が生ずる。

【0993】セレクト”ブレーキ”の実行は、内部的な例外”ブレーキ”を生じさせ、この例外”ブレーキ”は、オペレーション”while”によって、例えば、ブレーキテストステップ6524（図128）によって、またオペレーション”loop”、例えばブレーキテストステップ6808（図132）によって、またオペレーション”repeat”、例えばブレーキテストステップ7016

（図134）によってキャッチされる。前述した内部的な例外”継続”の表明の場合と同様に、内部的な例外”ブレーキ”が表明されると、オペレーション”while”、”loop”又は”repeat”の遂行の間の実行されるオブジェクトの反復的な遂行が終了される。しかし、内部的な例外”ブレーキ”の表明は、応答する実行されるオブジェクトのその後の反復的な遂行を表示させるのではなく、オペレーション”while”、”loop”又は”repeat”の遂行を終了させる。

【0994】内部的な例外”継続”及び”ブレーキ”は、オペレーション”while”、”loop”及び”repeat”のコンテキスト内においてのみ他の例外から識別されうる。従って、内部的な例外”継続”及び”ブレーキ”の以下の説明のコンテキストにおいて、オペレーション”while”、”loop”及び”repeat”は、集合的に、”関連のあるオペレーション”と呼ばれる。関連するオペレーションの各々は、内部的な例外”継続”及び”ブレーキ”を検出し、それに応答してアクションを行うので、内部的な例外”継続”及び”ブレーキ”は他の例外のように関連するオペレーションのどちらによっても表明されない。図135は例示的な例として用いられる。

【0995】スタック7104はプロセス7102のプロパティであり、プロセス7102の実行のスレッドを表している。ユーザーによって定義されたフレーム7106はスタック7104の底部にあり、オペレーション”live”の遂行の動的状態を記録する。底部から上部にかけてのスタック7104の残りの内容は、リピートフレーム7108、あらかじめ定義されたフレーム7110（オペレーション”loop”の遂行の動的状態を記録する）及びユーザーによって定義されたフレーム7112である。従って、図135のオペレーション内部は、”live”は、さらに別の1つのフィーチャの遂行を表示させるオペレーション”loop”の遂行を生じさせるオペレーション”repeat”の遂行を生じさせる。

【0996】ユーザーによって定義されたフレーム7112によってその動的状態を記録されるフィーチャの遂行が、セレクト”継続”又はセレクト”ブレーキ”を実行させると想定する。前述したように、セレクト”継続”又は”ブレーキ”の実行は、対応する内部的例外を表明する。これらの内部的な例外をキャッチするようにデザインされていないオペレーション、すなわち関連するオペレーション以外のオペレーションは、普通の例外が表明されたかのように挙動する（図109の例外としてステップ5210及び終了ステップ5212についての前記の説明参照）。この状態のもとでは、内部的な例外は、ユーザーによって定義されたフレーム7112によって表されるフィーチャの遂行によって伝搬され、前述したようにあらかじめ規定されたフレーム7110によって表されるオペレーション”loop”の遂行によって



キャッチされる。内部的な例外は、ループフレーム7110によって表されるオペレーション”loop”によって伝搬されず、従ってリピートフレーム7108によって表されるオペレーション”repeat”の遂行には影響しない。

【0997】関連のあるオペレーションは現に遂行されていない場合、すなわち、スタック7104に存在しない場合、セクタ”継続”又はセクタ”ブレーキ”が実行されると、クラス”ループ不在”の例外が発生する。クラス”ループ不在”の例外は、内部的な例外ではないので、他の例外と同様に、伝搬される。

【0998】内部的な例外”成功”が表明された場合、すなわちセクタ”成功”が実行された場合、現在の方法の遂行は成功の後に終了する。なお、あらかじめ規定されたフィーチャは直接に、すなわちエンジンを集合的に形成するコンピュータ命令中においてインプリメントされるので、ユーザーによって定義されたフィーチャのみが方法オブジェクトによってインプリメントされる。従って、その遂行の動的状態があらかじめ規定されたフレーム7110（図135）に記録されるオペレーション”loop”がセクタ”成功”を実行する場合、内部的な例外”成功”が表明され、オペレーション”loop”の実行が終了する。

【0999】オペレーション”loop”はあらかじめ規定されているので、オペレーション”loop”は内部的な例外”成功”をキャッチせず、従って内部的な例外”成功”を伝搬させ、その遂行の動的状態がリピートフレーム7108に記録されるオペレーション”repeat”を終了させる。同様に、オペレーション”repeat”はあらかじめ規定されているので、オペレーション”repeat”は内部的な例外”成功”をキャッチせず、その遂行の動的状態がユーザーによって定義されたフレーム7106中に記録されているオペレーション”live”を終了させる。しかしオペレーション”live”はあらかじめ規定されたインプリメンテーションを持たないので、オペレーション”live”は成功のうちに終了し、すなわち、例外の表明によって失敗に終わらない。同様に、ユーザーによって定義されたフレーム7112のサブジェクトを方法とするフィーチャが内部的な例外”成功”を表明した場合、すなわち、セクタ”成功”を実行した場合、そのフィーチャは、成功のうちに終了し、内部的な例外”成功”はキャッチされる。この場合、内部的な例外”成功”は、あらかじめ規定されたフレーム7110、リピートフレーム7108及びユーザーによって定義されたフレーム7106には影響しない。

【1000】前述したオペレーション、すなわち、オペレーション”do”、”if”、”either”、”while”、”select”、”catch”、”loop”及び”repeat”は、現在のメソッドのコンテキストにおいて遂行される。前述したように、あらかじめ規定されたオペレーシ

ョンは、方法オブジェクトによってインプリメントされない。前述したオペレーションは、すべてあらかじめ規定されており、従って方法オブジェクトによってインプリメントされない。前述した各オペレーションの、ユーザーのメソッドのコンテキストにおいての遂行は、一例として容易に記載される。

【1001】一例として、その方法がフレーム7112（図135）のサブジェクトであるフィーチャが、スタック7104のユーザーによって定義されたフレーム7112の直上のあらかじめ規定されたフレーム（図示しない）にその遂行が記録されている前記のあらかじめ規定されたオペレーションの1つの遂行をリクエストしたものと想定する。あらかじめ規定されたオペレーションの遂行の間に、ユーザーによって定義されたフレーム7112のプロパティ”stack”は、現在のスタックのままである。従って、あらかじめ規定されたオペレーションの遂行は、ユーザーによって定義されたフレーム7112のプロパティ”stack”に対して、アーギュメントをポップしたり、結果を保存したりする。従って前記のあらかじめ規定されたオペレーションは、現在の方法のコンテキストにおいて遂行される。

【1002】前述したオペレーション、すなわちオペレーション”do”、”if”、”either”、”while”、”select”、”catch”、”loop”、”repeat”、”continue”及び”break”は、実質的な一般性及び融通性を本発明によるオブジェクト志向命令セットに供与する。すなわち、エージェントは、1つのプレイスから別のプレイスに移動する間に複雑なロジックを使用し、いろいろなプレイスにおいて生み出されるエージェントとの相互作用によって情報をディポジットしたり収集したりすることができる。

【1003】すなわち、それ自身の運動をコンピュータネットワークを通じて差し向け、ネットワークを通じて複数の活性のそれ自身のコピーを送出し、遠隔のコンピュータノードに見いだされる他のコンピュータプロセスと相互作用し、それによってプロセスの間に情報を伝達することもできる新規なコンピュータプロセスのセットが提供される。新規なコンピュータプロセスのセットを形成するここに開示されたコンピュータ命令セットは、携帯型であり又一般的である。コンピュータ命令のセットはコンピュータネットワークの各々のノードに様にインプリメントされている。クラスは開示された命令セット内のオブジェクトとされる。本発明においてあるプロセスの命令は、コンパイルされるのではなく解釈される。コントロールコンストラクトは、リソースのマネージメント、決定の取扱い、例外の取扱い及びループコンストラクトのために用意される。従って、携帯型で一般的なコンピュータ命令セットが提供され、この命令セットからコンピュータプロセスの新規なセットが構成される。

【1004】コンピュータ命令の1つの特別のセットを以上に又アペンディックスに説明したが、本発明はここに記載された命令の機能性に限定されない。従って本発明の範囲は請求範囲の記載のみによって限定される。

【1005】アペンディックスA

著作権 ジェネラル・マジック社 1991、1992、1993。  
複製を禁ず。

【1006】以下の目次は、読者がアペンディックスの構成を理解しその情報を検索するために役立てるものである。

【1007】目次

1	序
1.1	テレスクリプト (Telescript) インストラクションセット
1.2	テレスクリプト エンジン
1.3	テレスクリプト ネットワーク
1.4	本アペンディックス
1.4.1	適用範囲 (Scope)
1.4.2	一致 (Conformance)
1.4.3	慣例 (Conventions)
1.4.4	構成 (Organization)
1.4.5	ロードマップ (Road Map)
1.4.6	リファレンス (Reference)
2	テレスクリプトの概念
2.1	モデル
2.2	オブジェクトモデル
2.2.1	オブジェクト
2.2.2	リファレンス
2.2.3	クラス
2.2.4	インヘリタンス
2.2.5	フィーチャ
2.2.6	属性
2.2.7	オペレーション
2.2.8	例外
2.2.9	コンストレイント
2.2.10	プロパティ
2.2.11	コピー
2.2.12	オブジェクトの初期化
2.2.13	オブジェクトの最終化
2.2.14	クラスの構造
2.3	実行モデル
2.3.1	メソッド
2.3.2	プロシージャ
2.3.3	実行されるオブジェクト
2.3.4	識別子
2.3.5	静的な置換の規則
2.3.6	動的な置換の規則
2.3.7	セレクトの実行
2.3.8	モディファイヤの実行
2.3.9	識別子の実行

2.3.10	メソッドの選択
2.3.11	メソッドの遂行
2.4	プロセスモデル
2.4.1	プロセス
2.4.2	位相
2.4.3	スレッド
2.4.4	リソース
2.4.5	パーミット
2.4.6	オーナーシップ
2.4.7	クローニング
2.4.8	ブランディング
2.4.9	コンタクト
2.4.10	アイソレーション
2.4.11	終了
2.5	ネットワーク・モデル
2.5.1	エージェント
2.5.2	ブレイス
2.5.3	トリップ
2.5.4	チケット
2.5.5	ミーティング
2.5.6	ペティション
2.5.7	占有
2.5.8	コンタクト
2.5.9	サイテイション
2.5.10	テレネーム
2.5.11	テレアドレス
2.5.12	相互変換
2.6	タイムキーピング・モデル
2.6.1	時間
2.6.2	カレンダー時間
2.7	パターンマッチングモデル
2.7.1	パターン
2.7.2	構造
2.7.3	他の非ターミナル
2.7.4	メタキャラクタ
3	テレスクリプトクラスの概観
3.1	グループ
3.2	カーネルグループ
3.2.1	クラス
3.2.2	コンストレインド
3.2.3	コンストレイント
3.2.4	例外
3.2.5	実行される
3.2.6	実行の例外
3.2.7	識別子
3.2.8	カーネル例外
3.2.9	マーク
3.2.10	モディファイヤ
3.2.11	ゼロ
3.2.12	オブジェクト

3. 2. 13	パッケージ	3. 6. 2	引用
3. 2. 14	プロシージャ	3. 6. 3	名前付け
3. 2. 15	プログラミングの例外	3. 6. 4	テレアドレス
3. 2. 16	有資格の識別子	3. 6. 5	テレネーム
3. 2. 17	リファレンスド	3. 7	プロセスグループ
3. 2. 18	セレクト	3. 7. 1	コンタクト
3. 2. 19	変更されない	3. 7. 2	コンタクティッド
3. 2. 20	予想されない例外	3. 7. 3	パーミット
3. 2. 21	検査	3. 7. 4	プロセスの例外
3. 3	プリミティブグループ	3. 7. 5	プロセス
3. 3. 1	ビット	3. 7. 6	リソース
3. 3. 2	ブーリアン	3. 8	エージェント及びプレイスグループ
3. 3. 3	ケースド	3. 8. 1	エージェント
3. 3. 4	キャラクタ	3. 8. 2	オーセンティケータ
3. 3. 5	整数	3. 8. 3	相互変換
3. 3. 6	数	3. 8. 4	手段
3. 3. 7	オクテット	3. 8. 5	プレイス
3. 3. 8	順序あり	3. 8. 6	チケット
3. 3. 9	プリミティブ	3. 8. 7	チケットスタブ
3. 3. 10	プリミティブの例外	3. 8. 8	トリップの例外
3. 3. 11	テレナンバ	3. 8. 9	移動しない
3. 3. 12	時間	3. 8. 10	ウェイ
3. 4	コレクショングループ	3. 9	ミーティンググループ
3. 4. 1	アソシエーション	3. 9. 1	ミーティングの例外
3. 4. 2	ビットストリング	3. 9. 2	ミーティングプレイス
3. 4. 3	コレクション	3. 9. 3	ベティション
3. 4. 4	コレクションの例外	3. 9. 4	ベティションド
3. 4. 5	コンストレインドディクショナリ	3. 10	その他のグループ
3. 4. 6	コンストレインドリスト	3. 10. 1	カレンダー時間
3. 4. 7	コンストレインドセット	3. 10. 2	種々の例外
3. 4. 8	ディクショナリ	3. 10. 3	パターン
3. 4. 9	ハッシュド	3. 10. 4	ランダムストリーム
3. 4. 10	レキシコン	3. 10. 5	実数
3. 4. 11	リスト	4	テレスクリプトクラスの詳細
3. 4. 12	オクテットストリング	4. 1	コンベンション
3. 4. 13	セット	4. 2	エージェント
3. 4. 14	スタック	4. 3	アソシエーション
3. 4. 15	ストリーム	4. 4	属性
3. 4. 16	ストリング	4. 5	オーセンティケイタ
3. 5	クラス定義グループ	4. 6	ビット
3. 5. 1	属性	4. 7	ビットストリング
3. 5. 2	クラス定義	4. 8	ブーリアン
3. 5. 3	クラスの例外	4. 9	カレンダー時間
3. 5. 4	フィーチャ	4. 10	ケースド
3. 5. 5	インプリメンテーション	4. 11	キャラクタ
3. 5. 6	インターフェース	4. 12	サイティション
3. 5. 7	メソッド	4. 13	サイティド
3. 5. 8	オペレーション	4. 14	クラス
3. 6	特定化グループ	4. 15	クラス定義
3. 6. 1	サイティション	4. 16	クラスの例外

- |       |                 |         |  |
|-------|-----------------|---------|--|
| 4. 17 | コレクション          | 4. 67   | クオリファイドアイデンティファイヤ  |
| 4. 18 | コレクションの例外       | 4. 68   | ランダムストリート  |
| 4. 19 | コンストレインド        | 4. 69   | 実数   |
| 4. 20 | コンストレインドディクショナリ | 4. 70   | リファレンスド  |
| 4. 21 | コンストレインドリスト     | 4. 71   | リソース   |
| 4. 22 | コンストレインドセット     | 4. 72   | セレクト   |
| 4. 23 | コンストレイント        | 4. 73   | セット  |
| 4. 24 | コンタクト           | 4. 74   | スタック   |
| 4. 25 | コンタクティド         | 4. 75   | ストリーム  |
| 4. 26 | ディクショナリ         | 4. 76   | ストリング  |
| 4. 27 | 例外              | 4. 77   | テレアドレス   |
| 4. 28 | 実行される           | 4. 78   | テレネーム  |
| 4. 29 | 実行の例外           | 4. 79   | テレナンバ  |
| 4. 30 | フィーチャ           | 4. 80   | チケット   |
| 4. 31 | ハッシュド           | 4. 81   | チケットスタブ  |
| 4. 32 | アイデンティファイヤ      | 4. 82   | タイム  |
| 4. 33 | インプレメンテーション     | 4. 83   | トリップエクセプション  |
| 4. 34 | 整数              | 4. 84   | アンチェンジド  |
| 4. 35 | インターチェンジド       | 4. 85   | アネクスペクティドエクセプション   |
| 4. 36 | インターフェース        | 4. 86   | アンムーブド   |
| 4. 37 | カーネルの例外         | 4. 87   | ベリファイド   |
| 4. 38 | レキシコン           | 4. 88   | ウェイ  |
| 4. 39 | リスト             | 5       | テレスクリプトシンタックス  |
| 4. 40 | マーク             | 5. 1    | テレスクリプト  |
| 4. 41 | 手段              | 5. 2    | キャラクタテレスクリプト   |
| 4. 42 | ミーティングの例外       | 5. 2. 1 | プリフエイズとコメント  |
| 4. 43 | ミーティングプレイス      | 5. 2. 2 | エグゼキュートオブジェクツ  |
| 4. 44 | メソッド            | 5. 2. 3 | 他の非ターミナル   |
| 4. 45 | その他の例外          | 5. 3. 1 | プリウエイズとコメント  |
| 4. 46 | モディファイヤ         | 5. 3. 2 | エグゼキューティドオブジェクト  |
| 4. 47 | ネームド            | 5. 3. 3 | 他の非ターミナル   |
| 4. 48 | ニル              | 5. 4    | 数値コード  |
| 4. 49 | ナンバ             | 5. 4. 2 | ブレデファインディドフィーチャーズ  |
| 4. 50 | オブジェクト          | 6.      | モジュールのシンタックス   |
| 4. 51 | オクテット           | 6. 1    | 一般的な構造   |
| 4. 52 | オクテットストリング      | 6. 2    | 詳細な構造  |
| 4. 53 | オペレーション         | 6. 2. 1 | モジュール  |
| 4. 54 | オーダード           | 6. 2. 2 | インターフェース   |
| 4. 55 | パッケージ           | 6. 2. 3 | フィーチャ  |
| 4. 56 | パターン            | 6. 2. 4 | コンストレイント   |
| 4. 57 | パーミット           | 6. 2. 5 | 他の非ターミナル   |
| 4. 58 | ペティション          | 7       | ブレデファインドモジュール  |
| 4. 59 | ペティションド         | 8       | ブレデファインドクラスグラフ   |
| 4. 60 | プレイス            | 1 序     |  |
| 4. 61 | プリミティブ          |         | このアペンディックスのこのセクションでは、テレスクリプト（以下、Telescriptをカタカナで表記する）・テクノロジーの3つの主要な要素を紹介する。：3つの主要な要素は、テレスクリプトインストラクションセット、すなわちこのアペンディックスの主題、インストラクションセットを実行する、テレスクリプト・エンジン |
| 4. 62 | プリミティブエクセプション   |         |  |
| 4. 63 | プロシージャ          |         |  |
| 4. 64 | プロセス            |         |  |
| 4. 65 | プロセスエクセプション     |         |  |
| 4. 66 | プログラミングエクセプション  |         |  |

及び相互接続するエンジンによって形成される、テレسكريプト・ネットワークである。

【1008】1. 1 テレسكريプトインストラクションセット

ここに“インストラクションセット”としてときには呼ばれる、テレسكريプトインストラクションセットは、適用分野を分散型システム及びアプリケーションにしようとするプログラミング言語を集成的に形成するコンピュータインストラクションの組である。インストラクションセットは、オブジェクト指向でインタプリタされるものである。マイクロフィルムのアペンディックス E、Fにおいて、インストラクションセットはときには“言語”とも言われる。

【1009】インストラクションセットの中に組み込まれる最も顕著なクラスは、クラス“エージェント (Agent)”及び“プレイス (Place)”である。エージェント、すなわちクラス“エージェント”のメンバであるコンピュータプロセスは、それ自身を検査変更し、それ自身をネットワーク中の一方のプレイスから他方のプレイスに転送し、そしてそこに見出される他のエージェントと相互作用することのできるアクティブなオブジェクトである。この能力は、特別な場合において特別なエージェントに対して特別な能力のみをプログラムまたは管理者のどちらか一方が許可できるようにする、パーミットによってバランスされる。プレイスすなわちクラス“プレイス”の一つのメンバであるコンピュータプロセスは、それ自身を検査変更することができ、エージェントに対する地域 (venue) 及びエージェントが相互作用するコンテキストとして取り扱う活動的なオブジェクトである。

【1010】あるエージェントは、あるプレイスに行き、そこでそのプレイス及びそのプレイスの他の占有者と相互作用する。エージェントとプレイスとはある距離においては相互作用できない。このようにリモートプログラミング (以下、単に Remote Programming を RP という) を実行するインストラクションセットは、より親しみのあるリモートプロシージャコーリング (以下、単に Remote Procedure calling を RPC という) パラダイムを実行するのではない。RP は、可能なシステムの要素をコミュニケーションなしに相互作用させることによって RPC を改善し、その待ち時間 (latency) を減少させることによって相互作用の兼ね合いを改善する。等価的には、RP は、システムの要素が、他の領域内に存在するそれ自身のエージェント—そしてこのようにそれら自身—によって他のものをあつらえる。

【1011】インストラクションセットは、以下に示すこれらの特徴を実現するようにしている。:

安全性

・ インストラクションセットは、その後者の許可 (permission) を得ることなしに他のプロセスと干渉し、ま

たはプロセスが実行中のコンピュータを直接操作する、その許容を越えたプロセスを防止する。これは、ビールの汚染を防止するのに役立つ。

【1012】携帯性

・ インストラクションセットは、ハードウェアの制約あるいはソフトウェアの制約に対して、または特殊なコンピュータシステムの特異性に対して何らの譲歩をしない。これによってネットワーク内部またはその回りのどこでもプロセスを実行することが可能となる。

【1013】拡張性

・ インストラクションセットは、インストラクションセットに組み込まれるものと同様の拡張 (stature) のオブジェクトをプログラマによって規定された情報の形式に対して与える。これによって特別の目的のためにインストラクションセットの拡張が可能となる。

【1014】向上

・ インストラクションセットは、volatile のストレージと非volatile のストレージとの間に区別を設けない。各々の情報オブジェクトは、本来、不変である。これによって抽象概念プロセスレベルを増大させ、プロセスのサイズを減少させる。

【1015】1. 2 テレسكريプト エンジン

インストラクションセットによって書かれたプログラムは、テレسكريプトエンジンによって生命がもたらされる。テレسكريプトエンジンは、ここでときに“エンジン”とも呼ばれるもので、エンジンは、プログラムが含むインストラクションを解読する。そのようなプログラムが、テレسكريプト (telescript) と呼ばれる。あるエンジンは、多くのテレسكريプトを同時に解読することができる。

【1016】あるエンジンは、エンジンが実行中のコンピュータのハードウェアまたはオペレーティングシステムに直接依存することなく、インストラクションセットの抽象概念を遂行することができる。エンジンは、基準 (platform) についてエンジンが必要とするコミュニケーション、ストレージ及び他のサブシステムへの標準的なインターフェースを使用してこれを行う。これらは、テレسكريプト コミュニケーション アプリケーション・プログラミング インターフェース (Telescript Communication Application Programming Interface: 以下、API という) を形成し、このインターフェースは、アペンディックス C として添付され、引用によって全体として本明細書の一部とされる。

【1017】エンジンは、インストラクションセットの抽象概念からの特別に許可されたエスケープ (privileged escape) を遂行することができる。これらのエスケープは、オペレーション、アドミニストレーション及びマネージメント (operational, administrative, and managerial: 以下、単に OAM という) のツールの、エンジン外の構成を可能にする。このようなツールは、大規模な

コミュニケーションシステムの成功にとって不可欠である。

【1018】1. 3 テレスクリプトネットワーク  
2つあるいは2つ以上のエンジンは、相互接続することができる。ここで、ときに”ネットワーク (Network)”と呼ばれる、結果としてテレスクリプトネットワークは、エージェントがその中で移動することができる宇宙である。ネットワークは、他のネットワークの一部ではなく、他のネットワークのクライアントと考えることができる複数のコンピュータシステムを含んでいる。

【1019】コンピュータシステムは、コンピュータシステムがエンジンを組み込んでおり、このようにエージェントが行き来できるプレイスを提供する場合にのみ、ネットワークの一部分となる。この要求は、情報の構造並びに伝送に関してネットワークを均質にする。

【1020】エンジンは、相互接続されており、その結果、エンジンは、エージェントをそれ自身の間に移動させている。エージェントが、アペンディックスBに記載されているテレスクリプトエンコード規則に従わせるこの目的のため、順番に並べたり、またはエンコードされる。結果としてオクテットストリングは、プラットフォームインターコネクトプロトコル (Platform Interconnect Protocol) によって規定されるようにエンジン間を移動する。プラットフォームインターコネクトプロトコルは、この開示の一部分としてアペンディックスFとして添付され、引用によって全体として本文の最初に組み込まれている。

【1021】1. 4 本アペンディックス  
このアペンディックスは、次のように説明される。

【1022】1. 4. 1 適用範囲 (Scope)  
このアペンディックスは、インストラクションセットのバージョン0. 8を規定する。テレスクリプトエンコード規則、テレスクリプト プロトコル及びテレスクリプトAPI は、すべてこのアペンディックスの範囲外にあり、それぞれアペンディックスB、F、Cの主題である。

【1023】インストラクションセットは、仮想的なマシンのためのインストラクションセットと見ることができる。この視点から、読者は、テレスクリプトを発生させるより高いレベルの言語を用いた、すなわちテレスクリプトを生成させるためのコンパイラであることを容易に描くことができる。カリフォルニア州マウンテンビューのジェネラル・マジック社は、ハイテレスクリプト

(High Telescript) と呼ばれるこのような言語を開発している。この開示のアペンディックスA-Fのコンテキストにおいて、インストラクションセット自体は、しばしばローテレスクリプト (Low Telescript) と呼ばれる。

【1024】1. 4. 2 一致 (Conformance)  
エンジンのメーカーは、このアペンディックスに対する一

致を適切に請求するためにはある要求を満たさねばならない。

【1025】1. 4. 3 慣例 (Conventions)  
以下に、このアペンディックス全体を通じて散文によるこれらの慣例が記載される。

【1026】・”the  $\alpha$  attribute” は、” その識別子が $\alpha$ に等しい属性” を意味する。

【1027】・” is a key” は、” キーに等しい” ことを意味し、” キーと同一である” ことは意味しない。

【1028】・” an X” は、一般に” Xの一つのメンバ” を意味し、” Xの一つのインスタンス” は意味しない。

【1029】・” throws X” は、一般に” Xの一つのメンバを送出する” ことを意味する。

【1030】・” whether statement” は、” statement iff true” を意味する。

【1031】・ iffは、数学においてそうであるように、” if and only if” を意味する。

【1032】・ MSBは、” most significant bit” を意味する。

【1033】・ LSBは、” least significant bit” を意味する。

【1034】注：このアペンディックス全体を通じて、注（この場合の注と同様のもの）は定義ではなく注釈である。

【1035】1. 4. 4 構成 (Organization)  
このアペンディックスは、8つのセクションに分けられる。セクション1は、この序である。セクション2は、インストラクションセットの主な概念を紹介する。セクション3は、インストラクションセットの予め規定されたクラスを概観する。セクション4は、それらを詳細に規定する。セクション5は、インストラクションセットの構文を規定する。

【1036】セクション6は、予め規定されたクラスへのインターフェースの形式的な定義に従われる慣例を規定する。セクション7は、これらの形式的な定義を包含する。セクション8は、ユーザ定義されたクラスはなく、予め規定されたクラスを包含するクラスグラフの一部を示す。

【1037】1. 4. 5 ロードマップ (Road Map)  
このアペンディックスの一部は、定義的であり、他の一部は、単に情報を与えるものにすぎない。定義を与えるセクションは、セクション1、2、4及び5である。対照的に、セクション3、7、8は、セクション4に示された事項の再構成であり、セクション6は、ハイテレスクリプトのいろいろな様相の再掲である。

【1038】異なる読者には、このアペンディックスの違った部分が必要になる。インストラクションセットの適用範囲及び構造のみに興味を持つ読者は、セクション1、2及び3のみを読めばよい。テレスクリプトプログ

ラムは、セクション4も必要である。ローテレスクリプトプログラマでは、キャラクターテレスクリプトをカバーするセクション5の部分もマスタすべきであるが、ハイテレスクリプトプログラマでは、その必要はない。ハイテレスクリプトコンパイラまたはエンジンのオペレータは、2進テレスクリプトをカバーするセクション5の部分が必要である。

【1039】テレスクリプトの実務者にとっては、セクション7、8は不可欠なリファレンスである。

【1040】1. 4. 6 リファレンス (Reference)  
このアペンディックスはこれらの他のドキュメントに依存する。

【1041】[10646]

Information technology -Universal Coded Character Set (UCS), ISO/IEC DIS 10646, International Organization for Standardization and International Electrotechnical Commission, 1990.

[ユニコード]

The Unicode Standard: Worldwide Character Encoding, Volume 1, Version 1.0, The Unicode Consortium, Addison-Wesley, 1991.

## 2 テレスクリプトの概念

インストラクションセットは、種々のコンセプトを規定し、インストラクションセットの最も重要な規定が、このアペンディックスのこのセクションに紹介されている。これらの概念は"モデル"に分けられる。サブセクションには、各モデルが割り当てられている。各サブセクション内で、より小さなサブセクションには、モデル中の各概念に振り分けられている。

【1042】インストラクションセットは、リモートプログラミング用であるから、言語、オペレーティングシステム及びネットワークの領域を網羅する多くの概念を含む。普通は別々であるこれらの領域は、このインストラクションセットにおいて一つにまとめられる。

【1043】2. 1 モデル

インストラクションセットの概念は、複数のモデルに分けられている。この分けられたモデルは、教育的な目的にのみ用いられ、いかなる意味においても遂行中には見られない。

【1044】つぎのモデルが規定されている。

【1045】オブジェクト

"object (以下、オブジェクトともいう)" モデルは、オブジェクト指向、例えばオブジェクト、リファレンス、クラス、オペレーション及び例外を規定する。

【1046】実行

"execution (以下、実行ともいう)" モデルは、逐次実行、例えばメソッド、プロシージャ及び識別子 (アイデンティファイヤ) を規定する。

【1047】プロセス

"process (以下、プロセスともいう)" モデルは、マ

ルチタスク処理 (multitasking)、例えばプロセス、リソース、パーミット、コンタクト及びオーナーシップを規定する。

【1048】ネットワーク

"network (以下、ネットワークともいう)" モデルは、ネットワークのアーキテクチャ、例えばエージェント、プレイス、トリップ、ミーティング及びテレネームを規定する。

【1049】タイムキーピング

"timekeeping (以下、タイムキーピングともいう)" モデルは、計時用、例えば時間及びカレンダー期間の計測のための手段を規定する。

【1050】パターンマッチング

"pattern matching (以下、パターンマッチングともいう)" モデルは、パターンマッチング、例えばパターンのための手段を規定する。

【1051】各モデルは、以上に列挙したと同じ順序で以下に提示される。各モデル内の概念は、論理的な順序で示される。

【1052】2. 2 オブジェクトモデル

このインストラクションセットは、このセクションが定義する"object model (以下、オブジェクトモデルともいう)" を実現する。

【1053】2. 2. 1 オブジェクト

オブジェクトは、情報と情報処理との両方のインストラクションセットの単位である。オブジェクトは、あるクラスの一つのインスタンスである。

【1054】注：オブジェクトはどんなものでもよく、例えばブーリアンのように単純なもの、または例えばディクショナリのように複雑なもののどちらでもよく、例えばストリングのように受動的なもの、または例えばプロセスのように能動的なものいずれか一方でよい。

【1055】注：メッセージ、すなわちインストラクションセットが理解されるアプリケーションで、オブジェクトには、メッセージのシステムの構成部分、例えばそのメールボックス及び分配リストと、システムが伝達する情報オブジェクト、例えばメッセージ及び配送レポートと、これらの情報オブジェクトの要素、例えばエンベロープフィールドが含まれる。

【1056】パーシステンス

各オブジェクトは"persistent (以下、パーシステントともいう)" である。エンジンが故障し、後に回復した場合、オブジェクトに対するただ一つの効果は、それが一時的に使用できなくなることである。

【1057】サイズ

オブジェクトが占有する近似的な不変の記憶量である"サイズ (size)" を有する各オブジェクトは、オクテット単位で測定される。

【1058】注：オブジェクトのサイズは、プレイス毎に変化していてもよい。



## 【1059】2. 2. 2 リファレンス

”reference (以下、リファレンスともいう)”は、オブジェクトが指示されまたアクセスされる手段である。

## 【1060】Protected vs Unprotected

リファレンスは、”unprotected (保護されていない)”または”protected (保護されている)”である。保護されていないリファレンスは、オブジェクトを変更することができ、保護されたリファレンスは変更することができない。保護されていないリファレンスは、保護されるようにできるが、逆は正しくない。

## 【1061】生成 (Creation)

各オブジェクトには、1つ以上のリファレンスが存在する。ある1つのオブジェクトの生成は、そのオブジェクトへの最初のリファレンスを作り出すことでもある。リファレンスは、オブジェクトが不変である場合にのみ保護される。追加のリファレンスは、希望するように生成することができる。生成されたリファレンスは、ソースリファレンスが保護されている場合にのみ保護される。

【1062】注：多くの予め規定されたオペレーションは、既存のオブジェクトへの新しいリファレンスを生成し、これらのリファレンスを結果としてリターンする。

## 【1063】比較 (Comparison)

オブジェクトに対するすべての保護されていないリファレンスは、同等である。あるオブジェクトに対する保護されていないリファレンスを用いて行われた変更は、そのオブジェクトに対して行われたもので、従って、実際にはそのオブジェクトに対する全ての他のリファレンスについて行われる。あるオブジェクトに対する全ての保護されたリファレンスも、相互に同等である。

【1064】保護されていないリファレンスと保護されているリファレンスとは、2つの点で異なっている。これらの相違点は、リファレンスが意味するオブジェクトのフィーチャを要求するために保護されたリファレンスが用いられる場合に現れる。第1に、フィーチャがオブジェクトを変更しようとする場合には、そのフィーチャは失敗し、”リファレンスが保護されている”を送出する。第2に、フィーチャが、コピーではないオブジェクトのプロパティの一つへのリファレンスをリターンした場合、そのリファレンスは保護される。

【1065】2つのリファレンスが同一のオブジェクトを意味するかどうかを定めることができる。

## 【1066】廃棄処理 (Discarding)

リファレンスが、必要とされなくなったときには廃棄すべきである。同一のオブジェクトに対するリファレンスが残存していない場合、オブジェクト自身は破壊される。

## 【1067】無効処理 (Voiding)

エンジンは、このアペンディックスの規定する事情のもとにリファレンスを無効にすることができる。”voided (無効にされたことを意味し、以下、ポイドされたとも

いう)”リファレンスは、もはやオブジェクトへのアクセスをしない。エンジンは、スタックにポイドされたリファレンスを保存 (push) するのではなく、”リファレンス ポイド (Reference Void)”を送出する。

## 【1068】2. 2. 3 クラス

”クラス”は、すべて同一のインターフェースと同一の遂行とを持つ、クラスの”インスタンス (instance)”であるオブジェクトのセットを規定する。クラス自身は、インターフェースと遂行との両方を持ち、これらは潜在的にクラスについてユニークである。

## 【1069】Predefined vs User-defined

クラスは、Predefined (予め規定されていることを意味し、以下、プリデファインドともいう) かまたはUser-defined (ユーザ定義されることを意味し、以下、ユーザデファインドともいう) のいずれかである。”プリデファインド”クラスは、インストラクションセットに組み込まれ、このアペンディックスによって定義されているもので、各々のテレスク립トプログラマにとって利用可能な一種のオブジェクトを表している。”ユーザデファインド”クラスは、プログラマによって定義されるもので、インストラクションセットを特定の目的のために拡張させる。

## 【1070】Concrete vs Abstract

クラスは、具体的でもあり、抽象的でもある。”concrete (以下、コンクリートともいう)”のクラスは、インスタンスを持つことができる。”abstract (以下、アブストラクトともいう)”のクラスは、インスタンスを持ち得ないが、そのサブクラスは持つことができ、またしばしば持っている。具体的なクラスまたはその遂行のスーパークラスの一つは、クラスに対する本来の各フィーチャまたはクラスによって引き継がれた各フィーチャを遂行する。

## 【1071】シーリング (Sealing)

通常、コンクリートであるクラスは、シールすることができる。エンジンは、シールされたクラスがユーザによって定義された直接のサブクラスを持つことを禁止するが、予め規定された直接のサブクラスを持つことは禁止しない。

## 【1072】クリエーション (Creation)

具体的なクラスは、新しいインスタンスを記述する初期パラメータが与えられる場合に、新しいインスタンスを生成することができる。しかしながら、各々のクラスの各インスタンスは、すべてこの方法で生成することはできない。残りのものは、最初にこのようにして一つのインスタンスを作り出し、次に必要に応じてインスタンスの属性を変更することによって作り出すことができる。

## 【1073】変更 (Conversion)

コンクリートなクラスは、あるクラスのインスタンスをそれ自身のインスタンスに変更することができる。しかしながら、各々のクラスの各インスタンスを各々の他の

クラスのインスタンスに変更することができるわけではない。オペレーション”コンバージョン(変更を意味し、以下コンバージョンともいう)”の定義と、コンバージョンが作り出すオブジェクトの性質とは、関係する2つのクラスに依存する。

#### 【1074】互換性 (Compatibility)

クラスは、一つの引用されたオブジェクトとして具体化される。あるクラスは、前者のインターフェースを他のインターフェースから、次の種類の変更を行うことだけで後者のインターフェースから生成することができる場合に他のクラスに対して後方に互換を有する。第1に、フィーチャを付加することができる。第2に、後者のインターフェースにおいて属性がリードオンリである場合には、その属性のクラスは、サブクラスにまで狭めることができる。第3に、現存しているオペレーションのアーギュメントのクラスは、一つのスーパークラスまで広げることができる。第4に、現存するオペレーションの結果のクラスは、一つのサブクラスまで狭めることができる。

#### 【1075】2. 2. 4 インヘリタンス

##### Native vs Inherited

予め規定されたものでもユーザー定義されたものでもよい、すべてのクラスは、”inheritance (引き継ぎを意味し、以下インヘリタンスともいう)”によって相互に対して関連付けられている。各クラスは、いろいろの特性を有する。一般に、クラス特性の内のあるものは、そのクラスに対して”native (固有を意味し、以下ネイティブともいう)”であり、他のものは、他のクラスから”インヘリタンス”されている。インヘリタンスは、移行的な関係である。あるクラスは、別のクラスから後者のクラスの引き継がれた特性だけでなく、後者の固有の特性も引き継ぐ。前記の用語を多少拡張すると、あるオブジェクトの固有の特性は、そのオブジェクトがインスタンスであるクラスに固有の特性である。

【1076】インストラクションセットは、クラスとその関係と簡単なグラフによって記述する、クラス間の引き継ぎ関係に対して十分に強制する。このグラフにおいて、ノードはクラスを表し、ノードの間の円弧は、それらのノードが表すクラスの間の引き継ぎ関係を表している。

#### 【1077】Subclass vs Superclass

あるクラスが別のクラスから直接または間接に特徴を引き継いでいれば、前者は後者の”subclass (以下、サブクラスともいう)”であり、後者は前者の”superclass (以下、スーパークラスともいう)”である。あるクラスが直接に別のクラスから特徴を引き継いでいれば、前者は後者の”immediate subclass (以下、イミディエートサブクラスともいう)”であり、後者は前者の”immediate superclass (以下、イミディエートスーパークラスともいう)”である。

【1078】あるクラスのインスタンスは、そのクラス及び複数のスーパークラスの”member (メンバ)”である。

【1079】フレーバあるクラスは、フレーバである。各々が抽象的であるかまたは具体的である”flavor (以下、フレーバともいう)”は、次のようにしてツリー構造を形成する。ツリーのルートは、クラス”オブジェクト”を表す。任意の与えられたソースノードから発するアーク (arc) によって到達される各々の目的点ノードは、そのソースノードが表すフレーバのイミディエートサブクラスである。従って、ソースノードは、各々のこのような目的点ノードによって表されるフレーバのイミディエートスーパークラスを表す。1つ以上のアークを次々に通って到達される各々の目的点ノードは、そのソースノードが表すフレーバのサブクラスを表している。このように、ソースノードは、各々のこのような目的点ノードによって表されるフレーバのスーパークラスを表している。

【1080】注：フレーバは、単一の受け継ぎを規定する。

#### 【1081】ミックスイン

他のすべてのクラスは、ミックスインである。”mix-in (以下、ミックスインともいう)”及び一すべてそれ自身ミックスインである—その0または1つ以上のスーパークラスは、次のようにして第2のツリーを形成する。このツリーのルートは、ミックスインである。アークは、ツリーのノードが表すミックスインの間の逆引き継ぎ関係を表している。ミックスインは、抽象的である。

【1082】注：ミックスインは、多重するインヘリタンスの限定された形式を規定する。

#### 【1083】クラスグラフ

フレーバとミックスインとは共に、ある指向されたグラフを形成する。グラフは、次の2つのステップで形成される。第1に、各ミックスインツリーのアークは、逆インヘリタンス関係ではなく、逆インヘリタンス関係を表すように再指向させられる。第2に、このように変更されたミックスインツリーは、フレーバツリーの上に重ねられる。

【1084】注：複数のクラスをグローバルに特定化することによって達せられるひとつのグローバルなクラスグラフがある。どんな特別なプレイスにおいても、グラフの知識が不完全であることがある。

#### 【1085】カノニカルオーダー

フレーバまたはミックスインであるクラスと、どちらもフレーバ及びミックスインであるそのスーパークラスとは、第3のツリーの道のりの順序である”canonical order (標準的な順序を意味し、以下、カノニカルオーダーともいう)”を有する。

【1086】ツリーは次のようにして構成される。そのルートは、問題のクラスであり、他のノードは、そのク

ラスのスーパークラスである。

【1087】アークは、ミックスインツリーと同様に、そのツリーのノードが表すクラスの間の逆引き継ぎ関係を表す。

【1088】道のりは、クラスが定義するその標準的な順序においてそのクラスのイミディエートスーパークラスが訪問を受ける第1の階層 (depth-first one) である。

#### 【1089】Interface vs Implementation

前述したすべては、クラス特性の3組いずれか、すなわちクラスのインターフェース、その導入 (implementation) またはその両方のどれについてもいえることである。このアペンディックス内のどの特定の点において、2組の特徴のどちらが説明されているかを明確にまたは暗黙の内に説明する。

【1090】注：クラスのインターフェースは、むしろ頻繁に説明されている。

#### 【1091】2. 2. 5 フィーチャ

”feature (以下、フィーチャともいう)” は、あるオブジェクトの外部から見られる特徴である。フィーチャは、オブジェクトが相互作用することを可能にする。あるオブジェクトが他のオブジェクトへのリファレンスを所有していれば、前者は後者のあるフィーチャを要求することができる。あるクラスのすべてのインスタンスは、そのクラスに固有の、またはそのクラスによって引き継がれたインターフェースによって定義される同一のフィーチャを有する。

【1092】あるフィーチャは、それが正しく使用されたときに”succeed (以下、成功ともいう)” し、他の場合には”fails (以下、失敗ともいう)” する。フィーチャが失敗する場合、例外が送出される。

#### 【1093】Attribute vs Operation

フィーチャは、次の二種類、すなわち属性及びオペレーションである。

【1094】注：ある特別なフィーチャが属性にされるかオペレーションにされるかは、部分的には趣味の問題である。

#### 【1095】シーリング (Sealing)

クラスは、固有のものか、または受け継がれたフィーチャを指定することができ、それによりユーザ定義されたサブクラスによるフィーチャの実行を禁止するが、予め定義されたサブクラスによるフィーチャの実行は阻止しない。

#### 【1096】実施 (Implementation)

クラスは、スーパークラスによって指定されないいかなる固有のフィーチャまたは引き継がれたフィーチャを実施することができる。後者の場合、クラスの実施は、もしあれば、スーパークラスの実施に取って代わるものである。

#### 【1097】アクセス (Access)

エンジンは、フィーチャへのアクセスを制御する。フィーチャの”access (以下、アクセスともいう)” は、フィーチャを使用し得るオブジェクトを定義し、次のもののいずれかに該当する。

#### 【1098】パブリック (public)

いかなるオブジェクトもこのフィーチャを使うことができる。

#### 【1099】プライベート (private)

そのフィーチャを有するオブジェクトのみがそれを使用できる。

#### 【1100】システム (system)

エンジンのみが、予め定義された、フィーチャを使用できる。

#### 【1101】2. 2. 6 属性

オブジェクトである”attribute (以下、属性ともいう)” は、あるオブジェクトが”requester (以下、リクエスタともいう)” では、別のオブジェクトか、または同一のオブジェクトかを要求し”responder (以下、レスポンドともいう)” では、別のオブジェクトか、または同一のオブジェクトかをおそらく取得し設定することのできるある特徴である。設定不可の属性は、”リードオンリ (read-only)” である。

#### 【1102】”αGet” vs ”αSet”

属性は、属性がリードオンリ、であれば、どれかひとつのオペレーションと正確に同等であり、リードオンリでなければ、両者である。”αGet” として象徴的に表される第1のオペレーションは、属性を取得する。それは、アーギュメントを持たず、その結果として属性を有する。”αSet” と呼ばれる、第2のオペレーションは、ある特定のオブジェクトで属性に変える。オペレーション”αSet” のひとつのアーギュメントは、そのオブジェクトであり、オペレーションは、結果として属性を持たない。

#### 【1103】Manual vs Automatic

クラスは、手動または自動によってある属性を実施することができ、そのクラスが抽象的である場合にのみ、全く実施しない。属性を”Manually (以下、手動的ともいう)” で実施することは、”αGet” の方法を規定しなければならず、属性が設定可能であれば、”αSet” である。属性を”Automatically (以下、自動的ともいう)” に実施することは、エンジンにそのようにさせることであり、エンジンは、属性がインスタンス属性であり、クラスが具体的であり、クラスのその実施スーパークラスも属性を実施しない場合に、そのようにする。エンジンは同一の識別子のプロパティで属性を遂行する。

【1104】インストラクションセットは、そのアーギュメントが属性の識別子 (”α”) である内部オペレーションである”getAttribute” 及び”setAttribute” の助けをそれぞれ借りて、属性の取得及び設定を定義する。これらの2つのオペレーションは、属性が自動的に

実施されていれば、その属性に関連されたプロパティをそれぞれ取得及び設定し、その他の場合、すなわち属性が自動的に実施されていれば、“ $\alpha$ Get”及び“ $\alpha$ Set”の各オペレーションについて、ユーザによって定義された方法を実施する。同様に属性の手動実施または自動実施は、オペレーションの実施の位置付けを行う実行モデルのアルゴリズムを使用して位置付けされる。

#### 【1105】格納位置 (Storage Location)

リードオンリでない属性は、デフォルトによって“Storage Location (以下、ストレージロケーションともいう)”として挙動する。すなわち、オペレーション“ $\alpha$ Get”は、もっとも最近に成功のうちに供給されたオブジェクトをオペレーション“ $\alpha$ Set”のアーギュメントとしてリターンする。このデフォルト挙動の例外は、特別の属性の散文の記載で呼び出されねばならない。

【1106】注：すなわち、デフォルト挙動を示す属性が、クラス“Dictionary (以下、ディクショナリともいう)”のひとつのメンバであるように制約され、クラス“Dictionary”のサブクラスのひとつのインスタンスが、オペレーション“ $\alpha$ Set”に供給される場合、オペレーション“ $\alpha$ Get”はそのクラスは、クラス“Dictionary”のインスタンスではなく、そのインスタンスをリターンする。

【1107】注：リードオンリであるすべての予め定義された属性は、デフォルト挙動を示す。

#### 【1108】2. 2. 7 オペレーション

オペレーションは、あるオブジェクト、すなわち“the requester (以下、リクエスタともいう)”が他のオブジェクト、または同一のオブジェクト、すなわち“the responder (以下、レスポンドともいう)”に実施するように求めることのできるすべてのタスクである。

#### 【1109】アーギュメント (Arguments)

リクエスタは、オペレーションの“arguments (以下、アーギュメントともいう)”である0またはそれ以上のオブジェクトをレスポンドに供給する。オペレーターは、オペレーションのアーギュメントの数が可変であるかまたは固定されているかを定め、後者の場合にはどのように各々が強制されるかを定める。

【1110】注：アーギュメントが、可変の数を持てば、いずれもマークとはならない。

【1111】注：予め定義されたオペレーションの中で、オペレーション“initialize”、“makeClasses”、“select”及び“new”は、可変数のアーギュメントを必要とし、他のいずれのオペレーションもそれを必要としない。

#### 【1112】リザルト (Result)

レスポンドは、オペレーションが成功すれば、リクエスタに、オペレーションの“result (以下、リザルトともいう)”である単一のオブジェクトをオプション的に供給することができる。あるオペレーションは、結果の有

無を定め、結果がある場合には、それがどのようにして強制されたかを規定する。

#### 【1113】2. 2. 8 例外

“exception (以下、例外ともいう)”は、ある特徴のための方法の遂行の失敗または、より基本的には、その方法のプロシージャ中の複数のアイテムである実行される複数のオブジェクトの内のひとつの実行の失敗を記述するオブジェクトである。

【1114】フィーチャは、もしフィーチャが成功した場合にレスポンドが、リターンするはずの結果の代わりに例外をレスポンドが“throws (投げ返す)”した場合に、失敗となる。フィーチャのリクエスタは、リクエスタが例外に反応することを望む場合に、その例外を“catch (受ける)”する。他の場合に、エンジンは、例外を“propagates (伝える)”させる。すなわち、エンジンは、リクエスタに例外を投出することを求める。

#### 【1115】2. 2. 9 コンストレイント

“constraint (以下、コンストレイントともいう)”は、コンストレイントの“subjects (以下、サブジェクトともいう)”である他のオブジェクトの上におかれる制限を規定する。可能な制限は、サブジェクトがメンバであるクラスの識別子、サブジェクトがそのクラスのインスタンスであるかどうか、そのサブジェクトが0であることが許容されるかどうか、上述したクラスにもかかわらず、そのクラス自身及びサブジェクトの通路を含む。

#### 【1116】パッセージ (Passage)

オブジェクトは、オブジェクトのソースがエンジンに伝えるソース リファレンスが、エンジンがオブジェクトの目的点に伝える目的点リファレンスを定めるいくつかの異なった方法のどれかひとつにおいて、フィーチャのリクエスタとレスポンドとの間に通されることができる。

【1117】オブジェクトの“passage (通路を意味し、以下パッセージともいう)”は、これらの識別子の内のひとつである。

#### 【1118】バイレフ (byRef)

目的点のリファレンスは、ソース リファレンスである。

#### 【1119】バイアンプロテクティドレフ (byUnprotectedRef)

目的点リファレンスは、ソース リファレンスであるが、エンジンは、ソースリファレンスが保護されている場合、“リファレンスプロテクティド (Reference Protected)”を投出する。

#### 【1120】バイプロテクティドレフ (byProtectedRef)

目的点のリファレンスは、ソースリファレンスがアクセスを与えるオブジェクトに対して、保護されたリファレンスである。

## 【1121】バイコピー (byCopy)

目的点リファレンスは、ソースリファレンスがアクセスを与えるところのオブジェクトのコピーに対する保護されないリファレンスである。エンジンは、この目的のためのみにコピーを作成する。コピーは、プロセスであれば、オブジェクトの目的点によって所有され、さもなければ、オブジェクトの目的点の所有である。

## 【1122】2. 2. 10 プロパティ

オブジェクトは、0またはそれ以上のプロパティを有する。それ自身がひとつの目的である”property (以下、プロパティともいう)”は、オブジェクトの内部状態のひとつの要素であり、全体としてみて、全内部状態を構成するオブジェクトのプロパティのすべてである。オブジェクトのプロパティは、オブジェクトに対して内部的であり、従って、そのオブジェクトの有無によって直接に感知され、検査され、変更されることができる。

【1123】オブジェクトのプロパティは、クラスによって区分される。0またはそれ以上は、オブジェクトのクラスに固有であり、0または1つ以上はクラスの実実施スーパークラスの各々に対して固有である。プロパティが、それに対して固有であるクラスの複数のメソッドによって、直接に感知され、検査され、変更することができるが、他のクラスまたはそのクラスのサブクラスもしくははスーパークラスに対して固有のメソッドによってはそのようにはできない。

## 【1124】2. 2. 11 コピー

オブジェクトはコピーすることができる。あるオブジェクトの”copy (以下、コピーともいう)”は、別のオブジェクトである。この別のオブジェクトは、”isSame”による場合ならびに、保護されたリファレンスがコピーを作成するために使用されたとしても、一般的に、そのコピーが変更され得ることによる場合以外は、コピー元のオブジェクトから区別できない。

【1125】あるコピーは、一般に、少なくとも最初は、そのコピーがコピー元になった、オリジナルと一般に同等である。”equal (同等を意味し、以下イコールともいう)”の意味は、オリジナルはメンバであるクラスに依存する。従って、クラスとコピーが不変であれば、コピーとオリジナルとは不特定の同等のままである。その他の場合には、コピーとオリジナルとは互いに独立であるため、相互に分岐していくことができる。

【1126】コピーは、オペレーション”copy”を用いて生成される。エンジンは、オリジナルの各プロパティのオペレーション”copy”をリクエストすることによってコピーを作成する。しかしながら、そのプロパティが、エンジンがこの目的のためにすでにコピーしたことのあるオブジェクトへのリファレンスであることを、エンジンが見出した場合には、エンジンはそのオブジェクトの第2のコピーを作らず、第1のコピーへのリファレンスを使用する。

## 【1127】2. 2. 12 オブジェクトの初期化

オブジェクトの生成は、オブジェクトの初期化を課する。オペレーション”initialize (以下、イニシャライズともいう)”の方法は、生成されたオブジェクトがそのメンバであるクラスに固有のものであり得る。方法は、そのクラスにとって固有のオブジェクトのプロパティを初期化することになる。

【1128】エンジンがオペレーション”イニシャライズ”の方法をリクエストする場合、エンジンは、その方法が、それ自体固有であるクラスにとって固有であるプロパティを0に設定する。

## 【1129】初期化の順序

オペレーション”initialize”に対する各々の方法は、その方法の責任範囲であるプロパティをイニシャライズした後に、オペレーションをエスカレートし、それにより他の方法が同じことをすることを可能にする。ある方法が、オペレーションをエスカレートすることなく成功した場合に、エンジンがその方法について同じことをする。エスカレーションのプロセスは、生成されたオブジェクトのクラスとそのクラスの遂行のスーパークラスを標準的な順序で訪問する。

## 【1130】初期化パラメータ (InitializationParameters)

あるクラスの”initialization parameter (以下、初期化パラメータともいう)”は、そのクラスにとって固有のオペレーション”initialize”の方法が、オペレーションの実施にとって必要とするオブジェクトである。もしクラスのミックスインであれば、クラスは、パラメータを選択することによってそれ自身ためのみに語る。クラスがフレーバであれば、クラスはそれ自身のみ及びその実施のスーパークラスのみについて語る。エスカレーションにおいて、オペレーション”initialize”のためのクラスの方法は、標準的な順序で後続するクラスのパラメータが最も上方のマークのさらに上においてスタックの上にあることを確実にするべきである。その他のクラスのための方法が実施された後、パラメータが残存していれば、クラス”Object”は、例外を投出する。

## 【1131】初期化の特権 (Initialization Privileges)

オペレーション”initialize”の方法は、エスカレーションの前ではなくエスカレーションの後に、生成されたオブジェクトのフィーチャを使用する。その場合にもオブジェクトは、メソッドがそれに対して固有であるクラスのインスタンスであるかのように応答するとき、オブジェクトがそのひとつのインスタンスであるクラスではない。メソッドは、どんな時にも他のオブジェクトのフィーチャを使用することができる。

## 【1132】2. 2. 13 オブジェクトの最終化

オブジェクトの破壊は、オブジェクトの最終化を課する。オペレーション”finalize (最終化を意味し、ファ

イナライズともいう) ”の方法は、破壊されるオブジェクトがそのひとつのメンバであるどのクラスにとっても固有とすることができる。方法は、そのクラスにとって固有のオブジェクトのプロパティを最終的なものにしなければならない。

【1133】オペレーション” finalize” のための方法は、そのオペレーションが最終化するプロパティをオペレーションによって廃棄し、したがっておそらくは破壊することができる。方法がそのようにしない場合エンジンがそれを行う。

【1134】最終化の順序 (Finalization Order) オペレーション” finalize” の方法は、オペレーションをエスカレートしない。エンジンは、破壊されたオブジェクトのクラス及びその実施スーパークラスを標準的な順序で訪問して、他の方法が成功しているか否かにかかわらず、各々の方法を実施する。

【1135】2. 2. 14 クラスの構造 インストラクションセットは、多くのコンピューター言語とは異なって、オペレーション” makeclass (以下、メイククラスともいう) ”によって実行中にクラスを生成することができる。あるクラスは、クラスのインターフェースとクラスの実施との両方を規定するクラス定義に基づいて生成される。

【1136】識別子のバインディング (Identifier Bindings)

クラスのインターフェースまたは導入は、そのインターフェースまたは実施が識別子によって指示する他のクラスに通常は依存する。サイテーションは、インターフェースまたは実施の属性” vocabulary” の助けを借りて各々のこのような識別子に対して拘束される。

【1137】識別子が、予め規定されたクラスの識別子に等しければ、サイテーションはそのクラスに等しい。その他の場合には、識別子が、属性” vocabulary” 中のあるキーに等しければ、サイテーションは、関連した値に等しい。その他の場合には、サイテーションの属性” title” が、そのクラスの識別子に等しいことを除いて、サイテーションは、構成されたクラスのサイテーションに等しい。

【1138】注：インストラクションセットは、予め定義されたクラスの割り当てられたサイテーションを定義しない。

【1139】サイテーションのバインディング (Citation Bindings)

クラスは、それぞれ現在のプロセス及び現在のプレイスの属性” privateClasses (以下、プライベートクラスともいう) ”及び” publicClasses (以下、パブリッククラスともいう) ”の助けをかりて、サイテーションに対して拘束される。引用されたクラスが予め定義されていれば、引用されたクラスは、エンジン自体に見い出される。もし引用されたクラスが、ユーザーによって定義さ

れていれば、そのクラスはサーチされ、現在のプロセスの属性” privateClasses” に見い出されるか、または見い出されず、見い出されない場合は、現在のプレイスの属性” publicClasses” に見い出され、そのクラスは、引用されたクラスに対して後方に互換性を有するどんなクラスでもよい。

【1140】クラスは、2回の内の1回において、サイテーションに対して拘束される。クラスの識別子があるプロシージャのアイテムである場合、そのクラスは、識別子が実行される都度サーチされる。その他の場合には、クラスが構成された場合にサーチされる。

【1141】2. 3 実行モデル

インストラクションセットは、このセクションが定義する” 実行モデル” を実現する。

【1142】2. 3. 1 メソッド

” method (以下、メソッドという) ” は、その各々の遂行の動的状態を別々に保持することのできるプロシージャである。この動的状態は、ひとつのフレームの形状をとる。オペレーション、変換及び選択された属性を遂行するためにメソッドが使用される。

【1143】フレーム (Frame)

” frame (以下、フレームともいう) は、スタックと0またはそれ以上の変数とを含む。それ自身がオブジェクトである” variable (変数) ” は、フレームのメソッドの特別な遂行の動的状態のひとつの要素であり、全体として集められかつスタックとともに、フレームのすべてのプロパティが、全体の動的状態を構成する。フレームのスタック及び変数は、そのフレームに対して内部的であり、従って、フレームのメソッドのみによって直接に感知され、検査され、変更することができる。フレーム自身は、全く操作され得ない。

【1144】遂行 (Performance)

メソッドを” performance (遂行を意味し、以下、パフォーマンスともいう) ” することは、ひとつのフレームを生成し、フレームのスタックの初期アイテムとしてオブジェクトの申し入れを受け入れ、フレームの変数を0に設定し、メソッドのプロシージャを実施し、フレームを廃棄し、プロシージャの成功した場合にのみフレームの最終的なアイテムを、フレームの廃棄前にそのスタックからそれらをポップして提供することである。プロシージャが成功するか、または失敗した場合にメソッドの実施は、それぞれ成功したり失敗したりする。

【1145】いかなる時点においても” current method (現在の方法) ” は、現在遂行中のメソッドであり、” current frame (現在のフレーム) ” は、その遂行のためにつくり出されたフレームであり、” current stack (現在のスタック) ” すなわち” the stack (以下、スタックという) ” は、現在のフレームのスタックである。

【1146】注：メソッドは、あるオペレーションを遂

行する場合、スタックの初期アイテムとして提供されたオブジェクトが、アーギュメントであり、もしあれば、最終的なアイテムとして提供されたオブジェクトの最も上方にあるものが結果である。これらの2つの事象の間で、スタックは、最初にアーギュメントを保持し、もしあれば、次にメソッドが要求するオペレーションの結果を保持する。もしあればオペレーションの終了時にスタックのもっとも上方にあるアイテム以外のアイテムが、オペレーションの中間のステップを実行するためにオペレーションによって遂行の間、使用されるオブジェクトである。

#### 【1147】2. 3. 2 プロシージャ

"procedure (以下、プロシージャともいう) は、プロシージャのプロパティに含まれず、より基本的に、プロシージャの一部分である、その" item (以下、アイテムともいう) "である、0またはそれ以上の実行されるオブジェクトを含む。これらのアイテムは、範囲 [1, n] によって番号付けされ、" n " は、プロシージャの" length (長さ) "である。プロシージャは、長さが0になれば" cleared (クリアされるという意味をなし、以下、クリアードともいう) "される。アイテムの番号は、プロシージャ中のその" position (位置) "である。" subprocedure (以下、サブプロシージャともいう) " は、一つのプロシージャ内の隣接した位置にある0以上のアイテムを含む。

#### 【1148】遂行 (performance)

プロシージャを" perform (遂行) "することは、現在のフレーム中において増大する位置の順序でそのアイテム順に実行されることである。プロシージャの遂行は、各々のアイテムが成功のうちに実行されれば" succeeds (成功) "し、他の場合には" fails (失敗) "する。いずれかのアイテムが実行に失敗した場合には、残りのアイテムは実行されない。

#### 【1149】2. 3. 3 実行されるオブジェクト

"executed (実行されるという意味をなし、以下、エグゼキューティドともいう) " オブジェクトは、ひとつのプロシージャのアイテムとして許可される。

#### 【1150】遂行

プロシージャではない実行されるオブジェクトを、" perform (遂行) "することは、それを実行することである。

#### 【1151】実行 (execution)

実行されるオブジェクト" execute (実行: 以下、エグゼキュートともいう) "することは、実行されるオブジェクトが識別子であるか、モディファイヤであるか、セクタであるか、または他の何かであるかということに依存して物事を行うことである。エンジンが例外を投出した場合には、実行は失敗し、他の場合には実行は成功する。

#### 【1152】識別子、モディファイヤまたはセクタ以

外のいかなる実行されるオブジェクトも、実行されるオブジェクトへの保護されるリファレンスをスタックに保存することによって実行される。

【1153】注: 前項において問題にされた実行されるオブジェクトは、ビット、ビットストリングブーリアン、文字、整数、マーク、オクテット、オクテットストリング、プロシージャ、実数及びストリングである。

#### 【1154】2. 3. 4 識別子

特別なクラス、フィーチャ、プロパティまたは変数は、インストラクションセット中の他の構成及びオブジェクトと同様に、識別子によって表される。

#### 【1155】識別子 (Identifiers)

" identifier (識別子) " は、長さが少なくとも1に等しいストリングである識別子の" text (テキスト) "によって、その範囲内において、あるオブジェクトを他のオブジェクトから識別する。

#### 【1156】有資格の識別子 (Qualified Identifier)

" qualified identifier (有資格の識別子) " は、あるフィーチャの0またはそれ以上の遂行を同一のフィーチャの0またはそれ以上の他の遂行から識別する。有資格の識別子は、その識別子のテキストによってフィーチャを表す。有資格の識別子は、有資格の識別子の" qualifier "であるクラスの識別子のテキストによって、遂行一すなわち、あるクラスに対して固有のまたはあるクラスによって引き継がれたもの一を表す。

#### 【1157】範囲 (Scope)

同一のスコープを持ついかなる2つの識別子も同一とはしない。

#### 【1158】クラス (Class)

インターフェースまたは遂行によって引用されるクラスは、その範囲が遂行のそのインターフェースによって引用される全てのクラスである識別子によって表される。

#### 【1159】フィーチャ (Feature)

あるオブジェクトのフィーチャは、そのオブジェクトが1つのインスタンスであるクラスによってクラスに固有か、またはそれによって引き継がれる全てのフィーチャを範囲とする識別子によって表される。

#### 【1160】プロパティ (Property)

あるオブジェクトのプロパティは、問題のプロパティそれ自身がそれに対し固有であるクラスに対して固有の全てのプロパティを範囲とする識別子によって表される。

#### 【1161】変数 (Variable)

あるフレームの変数は、そのフレームがその遂行に属するメソッドによって規定されたすべての変数を範囲とする識別子によって表される。

#### 【1162】シンタックス (Syntax)

クオリファイアのテキストは、表A. 1の文字を排除するものとする。

#### 【1163】

#### 【表1】



表 A. 1		
	キャラクタ	注
制御キャラクタ (Control Characters)		
	U+0000 ~ U+001F	アスキーでは
	U+007F ~ U+009F	上記と同様であるが、ビット7は0
スペース (Spaces)	各種	§ 4.2. の [ユニコード] U+0020, U+00A0等を参照
シンボル及びパunctuation (Symbols and Punctuation)	U+0021 ~ U+002F	アスキーでは
	U+003A ~ U+0040	アスキーでは
	U+005B ~ U+005E	アスキーでは
	U+0060	アスキーでは
	U+007B ~ U+007F	アスキーでは
	U+00A1 ~ U+00BF	ラテン1 (Latin1) では
	U+00D7 & U+00F7	乗算 ("×") 及び除算 (÷) の 符号
	U+2000 ~ U+2FFF	割当てられたシンボル
プライベートユースなキャラクタ (Private Use Characters)	U+E800 ~ U+FDFF	
特殊キャラクタ (Special Characters)	U+FFFO ~ U+FFFD	

テキストまたはクオリファイアの最初の文字は、表 A.  
2 の文字ならびに表 A. 1 の文字を排除するものとする。

【1164】

【表2】

表 A. 2		
	キャラクタ	注
10進数 (Decimal Digits)	各種	§ 4.1. の [ユニコード] U+0030 ~ U+0039, 等を参照
空きなしマーク (Non-Spacing Marks)	各種	§ 4.5. の [ユニコード] を参 照

注：1つのプログラムがハイテレスクリプトにある場合、そのテキストの最初の文字が下線 (" \_ ") (U+005F) である識別子を割り当てる特権は、ハイテレスクリプトコンパイラ自身に留保される。

【1165】2. 3. 5 静的な置換の規則 (Static Substitution Rules)

プロシージャは、下記の表 A. 3 の置換がなされたかのように遂行される。プロシージャ中の第1列にサブプロシージャが現出されたときは、いつでも第2列の第2行のサブプロシージャがそれに換えられる。

【1166】表 A. 3 は、テレスクリプトを定義するために用いられたノンターミナルを用いて各々のサブプロシージャを定義する。この表の第1セクションでは、" Identifier (識別) " は、それ自身かまたは " Q Identifier (Q識別子ともいう) " を表す。第2セクションでは " Identifier (識別子を意味し、以下、アイデンティファイアともいう) " はそれ自身のみを表す。

【1167】<sup>1</sup> この置換は、識別子がオペレーション " protect (プロテクトともいう) " の識別子でもオペレーション " ref (レフともいう) " の識別子でもない場合のみ適応される。

【1168】<sup>2</sup> この置換は、ひとつのアイデンティティである。

【1169】下記の表 A. 3 または表 A. 4 のもの以外のサブプロシージャ中に配置されたモディファイアを有するプロシージャを作成するように求められた場合、エンジンは例外を投出する。このアペンディックスのセクション3. 2. 12に説明される内部的なオペレーションは、表 A. 3 または表 A. 4 の置換のひとつのものの結果として要求された場合にのみ成功する。その他の場合には内部オペレーションにはアクセスすることができない。

【1170】

【表3】

表 A. 3	
サブプロシージャ (Subprocedure)	置換されるサブプロシージャ
SetAttribute Identifier	識別子宣言を "setAttribute" と交換
UseStack Identifier	UseStack Identifier <sup>1, 2</sup>
GetClass Identifier	"getClass" 自身の識別宣言
GetProperty Identifier	"getProperty" 自身の識別宣言
GetVariable Identifier	"getVariable" 自身の識別宣言
SetProperty Identifier	"setProperty" 自身の識別宣言
SetVariable Identifier	"setVariable" 自身の識別宣言
実行されるオブジェクト外の宣言	"ExpectedObject <sup>3</sup> " の宣言

### 2. 3. 6 動的な置換の規則

プロシージャは、表 A. 4 による置換が同様になされたかのように遂行される。表 A. 4 において "Identifier"

"r" は、それ自身かまたは "QIdentifier" を表す。

【1171】

【表 4】

表 A. 4	
サブプロシージャ (Subprocedure)	置換されるサブプロシージャ
Identifier	識別子宣言を "getAttribute" と交換
Demarcate Identifier	識別子の交換宣言の "getAttribute" との分離
Demarcate Identifier	識別子の分離 <sup>1</sup>

<sup>1</sup> この置換は、識別子が属性を示すときにのみ適応される。

【1172】<sup>2</sup> この置換、すなわちアイデンティティは、注 1 の置換が、すなわち識別子が属性を表さない場合にのみ適応される。

【1173】2. 3. 7 セレクタの実行  
"selector (セレクタともいう)" は、特別の実行効果を実現する。セレクタは、次のように実行される。

【1174】ブレイク (break)  
オペレーション "loop (以下、ループともいう)"、"repeat (以下、リピートともいう)"、または "while (ホワイルともいう)" を成功させる。どのオペレーションも遂行されていない場合には、"loop Missing (すなわちループ不在を意味)" が投出される。

【1175】クライアント (client)  
現在のクライアントへのプロテクトされないリファレンスをスタックに保存する。"current client (現在のクライアント)" は、リクエストがエンジン自身でない場合、例えばフィーチャのアクセスが "system (システムともいう)" である場合、現在のメソッドが、遂行しているフィーチャを要求するオブジェクトであり、その他の場合には、0 である。

【1176】継続 (continue)  
オペレーション "loop"、"repeat"、または "while" を行わせ、もし何かオペレーションがあれば、その次の遂行を開始させる。どちらのオペレーションも遂行されていないければ、"Loop Missing" が投出される。

【1177】エスカレート (escalate)  
現在のメソッドが、遂行するオペレーションのための識別子を実行するのと同じであるが、次のような例外がある。メソッド選択の前に、エンジンは、リクエストがレ

スポンダでない場合、"Escalation Invalid (エスカレーション不適切の意味)" を投出する。また、エンジンは、メソッドの選択の間、現在のメソッドが固有となるクラスをバイパスする。オペレーションは、エスカレートされている。

【1178】ここ (here)  
現在のプレイスへの保護されていないリファレンスをスタックに保存する。"current place (現在のプレイス)" は、現在のプロセスがエンジンプレイスでない場合に、現在のプレイスが占有されているプレイスであり、その他の場合には、0 である。

【1179】プロセス (process)  
現在のプロセスへのプロテクトされていないリファレンスをスタックに保存する。"current process (現在のプロセス)" は現在の方法の遂行が動的にその一部となっているプロセスである。

【1180】セルフ (self)  
現在のオブジェクトへのリファレンスをスタックに保存する。"current object (現在のオブジェクト)" は、現在のメソッドを遂行しているオブジェクトである。保護されるリファレンスが現在のメソッドの遂行をリクエストするために用いられている場合にのみ、リファレンスは保護される。

【1181】成功 (succeed)  
残りのアイテムを実行することなしに現在の方法を成功させる。属性、またはオペレーションの結果は、もしあれば、スタックに残される。

【1182】2. 3. 8 モディファイヤの実行  
モディファイヤはプロシージャにおいてそれに続くアイテムの実行を変更する。いろいろのモディファイヤの実行が置換規則に定められている。置換規則が関与しない

もののモディファイヤは、次のように実行される。

【1183】デマルケイト (demarcate)

次の識別子または有資格の識別子を実行させるが、次の点に変更される。スタックがマークを含まない場合、エンジンは、"Mark Missing (マーク不在の意味)" を投出する。識別子が表わすオペレーションによって要求されるアーギュメントの数が固定されており、マークよりも上方のオブジェクトの数よりも少なければ、エンジンは "Argument Invalid (アーギュメント不適切の意味)" を投出する。さもなければ、アーギュメントの数が固定されていてマークよりも上方のオブジェクトの数よりも多ければ、エンジンはスタック中に、マークと供給されるオブジェクトとの間に、不在のアーギュメントの数分の0を挿入する。エンジンは、次にスタックからマークを除き、オペレーションが成功しているが結果をもたらさない場合には、結果の代りにリクエストにゼロをリターンする。

【1184】メンション (mention)

"mention (以下、メンションともいう)" モディファイアに続くオブジェクトの実行を回避し、その代りにスタックにそのオブジェクトへの保護されたリファレンスを保存する。

【1185】ユーススタック (useStack)

次のオブジェクトを実行する前にスタック自身へのリファレンスをスタックに保存する。

【1186】2. 3. 9 識別子の実行

クラス、属性、プロパティまたは変数のための識別子の実行は、表A. 3及び表A. 4について以上に説明した置換規則によって定められている。

【1187】識別子

置換規則が関与しない識別子すなわち、オペレーションのための識別子一は、次のようにして実行される。

【1188】エンジンはスタックがクリアされた場合には、"Responder Missing (レスポнда不在の意味)" を投出する。それは、レスポндаがスタックの頂点にいたはずだからである。エンジンは、レスポндаが、識別子の表わすオペレーションを持たないかまたはリクエストがオペレーションへのアクセスを持たないときに、"Feature Unavailable (フィーチャ使用不可能の意味)" を投出する。その他の場合には、エンジンは、以下のセクションに述べるようにオペレーションのメソッドを選択し、実行する。

【1189】有資格識別子

オペレーションのための有資格の識別子は、そのオペレーションのための有資格の識別子でない識別子と同じ仕方で行われるが、次の例外がある。

【1190】メソッドの選択の前にエンジンは、リクエストがレスポндаでない場合に、"Escalation Invalid (エスカレーション不適切を意味し、以下、エスカレーションインバリッドともいう)" を投出する。さもなけ

れば、エンジンは、オペレーション "getClass (ゲットクラスともいう)" を要求し (そのアーギュメントは、テキストがクオリファイヤである識別子である)、オペレーションの結果が、現在のメソッドを固有とするクラスでも、その直接実施スーパークラスでもない場合に "Escalation Invalid (エスカレーションインバリッド)" を投出する。

【1191】エンジンは、メソッドの選択の間に、オペレーション "getClass" 及びその遂行スーパークラスの結果にそれ自身を限定する。オペレーションはエスカレートされている。

【1192】2. 3. 10 メソッドの選択

エンジンは、ここにリストされる順序で次の4つのローケーションをサーチすることによってそれが見出した最初のメソッドをあるオペレーションのために選択する。

【1193】・レスポндаがクラスであれば、そのクラスメソッド。

【1194】・レスポндаがクラスであれば、標準的な順序で探索されるその遂行スーパークラスのクラスメソッド。

【1195】・レスポндаのクラスのインスタンスメソッド。

【1196】・標準的な順序で探索されるレスポндаのクラスの遂行スーパークラスのインスタンスメソッド。

【1197】もしメソッドがオペレーションをエスカレートさせれば、エンジンは、別のオペレーションを選択することを試み、それが既に選択した方法が固有となるクラスに対するサーチを継続する。エンジンが方法を見出さなかった場合、エンジンは "Escalation Invalid" を投出する。

【1198】2. 3. 11 メソッドの遂行

エンジンはオペレーションのためにそれが選択したメソッドを次のように遂行する。

【1199】アーギュメントの数の固定

オペレーションが固定された数のアーギュメントを必要とする場合、エンジンは、スタックの長さがこの数字プラス1よりも少ない場合に、"Argument Missing (アーギュメント不在を意味し、以下、アーギュメントミッシングともいう)" を投出し、アーギュメントがアーギュメントのコンストレイントを侵害している場合に、"Argument Invalid" を投出する。

【1200】アーギュメントの数の変動

オペレーションが可変数のアーギュメントを必要とする場合、エンジンは、スタックがマークを含まない場合に "Mark Missing (マーク不在を意味し、マークミッシング)" を投出し、アーギュメントがアーギュメントのコンストレイントを侵害している場合に、"Argument Invalid" を投出する。

【1201】遂行 (Performance)

エンジンは、スタックからレスポндаを復元し、メソッ

ドの遂行のためのフレームを生成し、そのメソッドを遂行し、フレームを廃棄する。

【1202】以下の動作は、メソッドの遂行を取り巻いている。エンジンは、オペレーションのアーギュメントを提供し、リクエストのスタックからレスポンドのスタックにそれらのアーギュメントを移動させる。アーギュメントの数が可変の場合には、マークは、同様にこのようにして提供される。エンジンは、もしあれば、オペレーションの結果の提供を受け入れ、それをレスポンドのスタックからリクエストのスタックに移動させる。

#### 【1203】成功 (Success)

メソッドが、成功しオペレーションがある結果を生じるべき場合には、エンジンは次のことを行う。結果が生じない場合エンジンは”Result Missing (結果不在を意味し、以下、リザルトミッシングともいう)”を投出する。結果が結果のコンストレイントを満たさない場合、エンジンは”Result Invarid (結果不適切を意味し、以下、リザルトインバリッドともいう)”を投出する。どちらの例外も、次のように取り扱われる。

#### 【1204】失敗 (Failure)

メソッドが失敗し、オペレーションがその投出するべき例外を投出しなかった場合、エンジンは”Unexpected Exception (予期されない例外を意味し、以下、アンイクスペクティドイクセプションともいう)”によって、前記の例外を代える。

#### 【1205】2. 4 プロセスモデル

インストラクションセットは、このセクションが定義する”プロセスモデル (process model)”を実現する。

#### 【1206】2. 4. 1 プロセス (Processes)

”プロセス (process)”は、ネームされた自律的な計算である。それは、マルチタスクのためのインストラクションセットの規則を構成する。

【1207】注：あるプロセスは、実世界における何物か、例えば、ネットワーク中においてプロセスがその目的を促進しようと努めている人をしばしば表わす。

【1208】注：2つの種類のプロセス、すなわちエージェントとプレイスがある。

#### 【1209】相互作用 (Interaction)

プロセスが、別のプロセスへのリファレンスを持つ場合、両者は、2つの任意のオブジェクトが行い得るものと同じ仕方でも相互作用をすることができる。すなわち、これは、それぞれのフィーチャによる。このフィーチャの各特性または結果は、あるオブジェクトへのリファレンスまたはそのコピーを一方のプロセスが他のプロセスに伝達すること可能にする。

#### 【1210】2. 4. 2 位相 (Phases)

プロセスは、複数の位相を通過する。エンジンはオペレーション”イニシャライズ”を要求し、その時の可能なプロセスは、そのオペレーションを成功のうちに遂行する。エンジンはオペレーション”live (以下、ライブと

もいう)”を要求し、プロセスはそのオペレーションを成功のうちに遂行する。オペレーションのアーギュメントとしてゼロが供給される。エンジンはオペレーション”finalise (最終化)”を要求し、プロセスはそのオペレーションを遂行する。

#### 【1211】例外 (Exceptions)

前記の各位相の通常の進行状態は中断させることができる。オペレーション”initialize”が例外を投出した場合、エンジンはそのプロセスを終了させる。オペレーション”live”が例外を投出した場合、そのプロセスが保持するパーミットがそれを許すならば、オペレーション”live”を再び要求することによってプロセスを再開するかまたはさもなければオペレーション”finalize”を要求する。前者の場合、例外はオペレーションのアーギュメントとして供給される。オペレーション”finalize”が例外を投出する場合、エンジンはプロセスを終了させる。

【1212】プロセスが前記の位相のどれかにおいてその固有のパーミットまたはローカルパーミットを使い尽くした場合、プロセスが”Permit Exhausted (パーミット消尽を意味し、パーミットイクスハウステッドともいう)”を投出したかのような状態となる。

#### 【1213】2. 4. 3 スレッド (Threads)

各プロセスは、独立した実行スレッドを生じさせる。この実行スレッドは、どんな時点においても、プロセスのイニチアティブにおいて遂行されているメソッドのフレームを包括する。第1のこのようなメソッドは、プロセス自身がオペレーション”live”のために供与する方法であり、第2のメソッドは、オペレーション”live”の方法がそのオブジェクトのために要求するフィーチャのためにオブジェクトが用意する方法であり、以下同様であり、反復される。

#### 【1214】スケジューリング (Scheduling)

エンジンは、ブロックされていない、すなわちある時間まで待機しているすべての実行スレッドをスケジュールし、そのプライオリティによって必要とされる優先順序に従って行う。他のものが等しければ、エンジンは、1つのオーソリティのプロセスを他のオーソリティのプロセスに対して優先させることはなく、またひとつのブランドのプロセスを他のブランドのプロセスに対して優先させたり、同一のプライオリティのあるプロセスを別のプロセスに優先させたりすることはない。

#### 【1215】中断 (Interruption)

エンジン自身があるオブジェクトのフィーチャを、すなわちシステムフィーチャを要求する場合、そのフィーチャのメソッドは、オブジェクトのオーナーのスレッドの一部として遂行される。オーナーは実際には、その目的に対しては中断される。

#### 【1216】2. 4. 4 リソース (Resources)

”resouce (リソースともいう)”は、ある特別のプロ

シーの遂行を通じてあるプロセスに対してある保証を与えることができる何かである。エンジンは、これらの保証を与えることができるまでプロセスを遅延させることができる。

【1217】注：リソースは、臨界的な条件領域の定義を可能にする。

【1218】注：リソースは、物理的なリソース（以下に例示する）のための代理人であることが多い。

【1219】注：電子メールシステムにおいて、例えばメールボックスとして役立つエージェントが送出されるメッセージのためのデータベース及びこれらのデータベースを表わすリソースをそのプロパティの中に含めることができる。エージェントは、新しく送出されるメッセージをデータベースに追加するタスクを開始する前にリソースの排他的な使用を得ることがあり、それによって現在の配送に対してデータベースの一体性を確保する。

【1220】ニード (Need)

プロセスは、別のプロセスに、あるオブジェクトへのリファレンスを伝えることができるので、各オブジェクトは、そのクラスに固有のまたはそのクラスによって受け継がれたメソッドがこれらのプロセスによって同時に遂行されることに対して準備しなければならない。リソースは、オブジェクトがこれらの準備をすることを可能にする。

【1221】リソースは、オペレーションの連鎖が細かく (Atomically) 行われることを保証するために用いられる。あるオペレーションに対してはこのような注意は一般には必要ではない。それは、オペレーション "wait" (ウェイトという) 、オペレーション "meet" (ミートという) 及びレスポンス例えばオペレーション "do" (ドゥーという) としてかまたはアーギュメント例えばオペレーション "restrict" (制限を意味し、レストリクトという) として供給されたプロシーダを遂行するすべての予め定義されたオペレーションは、エンジンによって細かく遂行されるからである。

【1222】注：エンジンは、オペレーション "go" 及び "send" をアトミックに遂行する。

【1223】排他的と共有 (Exclusive vs Shared)

あるプロセスには、リソースの共有された使用または排他的な使用が保証される。どんな時点においても、いかなるプロセスもリソースの使用を持たないか、あるプロセスがリソースの "exclusive" (排他的なという意味を有し、イクスクルーシブともいう) "使用を持つか、または1つ以上のプロセスがリソースの "shared" (共有されたという意味を有し、シェアードともいう) "使用を持つかである。

【1224】条件 (Condition)

要求がなされたときに特定される1つ以上の条件の中にリソースが含まれることがプロセスにおいて保証される。いかなる時点においても、リソースは、そのリソー

スが生成されたときに定義された1つ以上の条件の中に、"condition (条件)" を持つ。識別子によって表わされるリソースの条件は、リソースの使用、この場合、共有でも排他的でもよい、によってのみ検査でき、そしてリソースの排他的な使用を持ったプロセスによってのみ変更され得る。

【1225】2. 4. 5 パーミット (Permits)

"permit (パーミットともいう)" は、あるプロセス、その "subject (サブジェクトともいう)" に能力を許容する。これらの能力は、インストラクションセットの大部分例えばインストラクションセットの算術演算によって表わされ、各々のプロセスに暗黙のうちに許可されている。他の能力は、特別のプロセスに対して、これらのプロセスに割り当てられたパーミットによって明文で許可されている。

【1226】注：パーミットの主要な目的は、コンピュータ及びコミュニケーションのリソースを予定されない量においてプロセスが消費することをできるだけ阻止することにある。これは、プロセスを生成してそのプロセスに対してお金を支払うかもしれないユーザにとっても、プロセスが消費するリソースを供給しなければならない供給者にとっても利益である。

【1227】注：前述したリソースは、クラス "resource (リソース)" のメンバと混同されるべきではない。

2つの間には必要な関連性はない。

【1228】能力 (Capabilities)

パーミットは、ある限られた大きさ及び量において能力を許容する。

【1229】パーミットは、プロセスにあるエイジ、サイズ、料金、優先、または真正 (オーセンティシティともいい; authenticity) を許容することができる。パーミットが、そのサブジェクトに対して資格を与える料金は、サブジェクトの "allowance" (以下、アローワンスともいう) と呼ばれる。これらのディメンションのいずれにおいても量は、整数である。エイジは秒とし、サイズはオクテットで、また料金は "teleclick" (以下、テレクリックともいう) で計測される。優先度はその整数の値と共に増大する。オーセンティシティは、領域特異である。

【1230】パーミットは、プロセスに、それ自身を搬送する権利 (パーミットの属性 "canGo") 、それ自身のクローンを作り出して搬送する権利 (属性 "canSend") 、他の競争プロセスを作り出す権利 (属性 "canCreate") 、選定されたプロセスのパーミットを、それらの能力を減少させる (属性 "canDeny") または増大 (属性 "canGrant") させるように選定されたプロセスのパーミットを設定する権利、または、他のプロセスの実際の料金を増大させる権利 (属性 "canCharge") 、または終了された時に再スタートさせる権利 (属性 "canRestart") を許容することができる。これらのディメ

ンションのどれにおいても量はブーリアンであり、このブーリアンの値が“真”であれば、問題の権利は許可される。

【1231】パーミットが能力を許容することのできる各ディメンションは、そのパーミットの属性によって表わされる。属性は、通常は整数であるが、その代りにゼロであることもでき、その場合に能力は無限の量で許容される。

【1232】注：エンジンによってもまたは他のプロセスによっても、あるプロセスに与えられるサービスについてのテレクリックで表わした料金は、プレイスごとにまたは時間ごとにまたはその両方について変化し得る。あるサービスに対するチャージ、例えばスペースは、単位時間ごとに評価することができる。

【1233】ステイタス (Status)  
あるプロセスの“status (以下、ステイタスともいう)”は、そのプロセスの行使された能力を特定する。あるプロセスのステイタスは、以下に説明する、そのプロセスの実効的なパーミットであるが、パーミットの“age (エイジともいう)”、“charge (チャージともいう)”、“extent (範囲の意味)”及び“priority (優先度の意味)”の各属性は、それぞれそのプロセスの実際のエイジ、チャージ、大きさ及び優先度である。

【1234】実効的パーミット (Effective Permit)  
プロセスの“effective Permit (実効的パーミット)”は、そのプロセスの許容された能力を特定する。このパーミットは、どんな時点及びプレイスにおいても、そのプロセスの本来の、領域的な、そしてローカルなパーミットと、効力を持つ時間的なパーミットとの間の交点である。しかしこの目的のために、本来のパーミットの“オーセンティシティ (authenticity)”属性は、プロセスの生成された領域にそのプロセスがない場合には、ゼロと考えられる。

【1235】2つのパーミットの“交点 (intersection)”は、それ自体としてパーミットであり、このパーミットの本来の属性のおおの、 $A_0$ は、問題の2つのパーミットの同じようなネームの属性 $A_1$ 及び $A_2$ の交点である。より特定的には、 $A_0$ は、 $A_2$ がゼロである場合に、 $A_1$ であり、 $A_1$ がゼロである場合 $A_2$ であり、その他の場合には、 $A_1$ と $A_2$ とのどちらか最小のものである。一旦設定されたプロセスのパーミットのどれかが再設定された場合には、エンジンは、後述するオペレーション“restricted (制限される)”の使用によってプロセスにそれを通知する。

【1236】プロセスは、プロセスの言う実効的なパーミットによってプロセスに許可される量以上の量において能力を行使しようと努めた場合に、そのパーミットを“violates (侵害)”している。パーミットを使い尽くすことなしに自己のパーミットを侵害すると、例外が引き起こされる。

【1237】プロセスの一時的なパーミットのどのひとつも行使されていない場合、プロセスの実効的なパーミットがプロセスに許容すべきエイジ、チャージ、またはサイズにプロセスの実際のエイジ、チャージまたは大きさがそれぞれ到達することを許容することによって、プロセスのパーミットが“消尽 (exhausts)”される。自己のパーミットの消尽は自己の終了を開始させる。

【1238】固有のパーミット (Native Permit)  
あるプロセスの“native permit (固有のパーミット)”はプロセスの創始者によってプロセスに許容された能力を特定する。あるプロセスすなわち親は、別のプロセスすなわち子を親が作ったときにこのパーミットを設定する。同僚プロセスは同僚の自身の実効的パーミット (属性“canDeny”) によってそのように能力が与えられるならば、能力を減少させることができるが増大はさせることができない。

【1239】能力はプロセスの間に転送させられるが生成はされない。親または同僚は、親または同僚がそれ自身持たない能力“X”を子に許容することはできない。その実効的パーミットによって親または同僚に許可された“X”の量を“P”で表わすものとする。さらに、子の固有のパーミットによって子に許容された“X”の量を“C”で表わすものとする。“C”は“P”を超過しない。

【1240】チャージ能力“X”は保存されている。親または子の実際のチャージの量を予め“A”とする。“C”は“P-A”を超過しない。親または同僚の実際のその後のチャージの量は“A+C”である。

【1241】地域的なパーミット (Regional Permit)  
プロセスの“regional permit (地域的パーミットを意味し、リージョナルパーミットともいう)”は、プロセスの占有するプレイスを含む領域によってプロセスに許可された能力を特定する。領域のオーソリティのプレイスは、プレイス自身の実効的パーミット (属性“canDeny”及び“canGrant”) によってそのように能力が保護されていれば、プロセスが領域に入る時及びその後のどの時点においても、パーミットを設定することができる。しかし、子がオペレーション“go”または“send”ではなく、オペレーション“new”を用いてエンタした場合には、この最初の地域的なパーミットは、親の現在の地域的なパーミットにされる。

【1242】ローカルパーミット (Local Permit)  
プロセスの“local Permit (ローカルパーミットともいう)”はプロセスが占有するプレイスによってプロセスに許可される能力を特定する。プレイスは、プレイス自身の実効的パーミット (属性“canDeny”及び“canGrant”) によってそのように能力が賦与されていれば、プロセスがエンタした時及びその後のどの時点においても、パーミットを設定することができる。

【1243】あるエージェントは、チケットを用いてあ

るプレイスにエンタした場合、そのプレイスのある能力をリクエストする。しかしながら、プレイスは、エージェントが持つべき能力を定める。そのプレイスは、エージェントが要求したよりも多くまたは少ない能力のローカルパーミットをもってエージェントがエンタすることを許容することができる。しかし、後者の場合にはトリップは失敗に終るすなわちオペレーション” go” または” send” は、エージェントが到着したとしても例外を投出する。

【1244】あるプロセスは、あるプレイスにそこで作り出されることによってエンタした場合、初期化パラメータなしに作り出されたローカルパーミットをプロセスの最初のローカルパーミットとして受ける。

【1245】一時的なパーミット (Temporary Permits)

プロセスの0またはそれ以上の” temporary permits (一時的なパーミットと意味し、テンポラリーパーミッツともいう)” は、プロセスが一時的にそれ自身に許容する能力を特定する。おのおのは、特定されたプロシージャの遂行についてオペレーション” 制限 (restrict)” または” スポンサー (sponsor)” を用いて実行される。これらのプロシージャは入れ子させることができる。

【1246】エンジンが、あるシステムのフィーチャをリクエストする場合、レスポンドのオーナーは、イニシャライゼーションパラメータなしに作り出された一時的なパーミットを用いてオペレーションのメソッドをスポンサする。このようにしてチャージは、オーナーに帰せられる。

【1247】注：あるプロセスは、一時的なパーミットを使用して、そのプロセスの許容されたエイジ、チャージまたはサイズの一部分を保存することができる。プロセスがこの品目の大部分を消滅した場合、プロセスは保存されている部分を使用して緊急アクションを取ることができる。

【1248】2. 4. 6 オーナーシップ  
それ自身プロセスでないおのおのオブジェクトは、オブジェクトによって” owned (所有)” されている。あるプロセスとそれが所有するオブジェクトとは、そのプロセスの” artifacts (以下、アーティファクトともいう)” である。

【1249】注：オーナーシップは、オーソリティと混同されるべきではない。

【1250】オブジェクトの生成

各オブジェクトは、最終的に、あるフィーチャのアーギュメントまたはフィーチャの結果をメソッドの間のコピーによって通過させることによって生成される。新しいオブジェクトすなわちパスされるコピーは、オブジェクトがプロセスである場合、コピーがそのメソッドに伝えられるオブジェクトによってかまたはその他の場合には、オブジェクトのオーナーによって所有される。

【1251】オブジェクトの破壊

オブジェクトが破壊される時、そのすべてのアーティファクトは破壊され、すべての残存するそれらへのリファレンスはボイドされる。

【1252】オブジェクトの共有

プレイスのアーティファクトは、第1の領域及びそれ自身のアーティファクトを形成する、直接スーパープレイスのアーティファクト並びに集合的に第2の領域を形成する、その直接サブプレイスのアーティファクトを意味するが、エンジンは、これら2つの領域の間にパスされたりファレンスをボイドすることができる。

【1253】注：このようにして防火壁をプレイスの間に設立することができる。

【1254】2. 4. 7 クローニング

あるプロセスには、それ自身をクローニングする能力が与えられることがある。クローンは、別のプロセス、すなわち” オリジナル (original)” のコピーとして開始されるプロセスであるが、次のような例外がある。

【1255】クローンのアイデンティティ (オーソリティではない) はオリジナルのアイデンティティとは異なっている。

【1256】クローンの固有のパーミット (そのローカルパーミットでもある) はクローンが作り出された時に与えられるパーミットである。オリジナルにおいて継続中の、すなわちオペレーション” 制限 (restrict)” の結果としての一時的なパーミットは、クローンの固有のパーミットがこれを必要とする限りにおいて、クローン内においてより制限的にされる。

【1257】オリジナルがそれ自身にまたはオリジナルが所有するオブジェクトに関するものである場合、クローンはそれぞれそれ自身に関するかまたはそのオブジェクトのそれ自身のコピーに関する。オリジナルな第3のプロセスのアーティファクトに関する場合、そのクローンはボイドされたりファレンスを有する。

【1258】オリジナル従ってクローンがコンタクトされるオブジェクトである場合、クローンの属性” コンタクト (contacts)” はクリアされる。

【1259】2. 4. 8 ブランディング

ある領域は、その内部の各プロセスに対するブランドである。

【1260】ある領域内のゼロまたは1つ以上のプロセスによって作り出された識別マークである。エージェントがある領域に入る時、その領域は、そのエージェントに新しいブランドを与える。その領域内のあるプロセスが別のプロセスを作り出す場合、後者のプロセスは前者のプロセスのブランドを担う。従ってブランドは、その領域に入ったプロセス例えばエージェントと、前者のプロセスによってそこで作り出された全てのプロセスとを意味する。

【1261】注：ブランドは、それ自身がインストラク



ションセットにないトラッキング・メカニズムを可能にする。

【1262】2. 4. 9 コンタクト (Contacts)  
 "contact (コンタクトともいう)" は1つのプロセスすなわち "observer (オブザーバ)" のために、第2のプロセスすなわち "subject (サブジェクト)" との相互作用を表し且つドキュメントする。"contacted (コンタクトされた)" プロセスは、そのコンタクトのセットを保持する上に、エンジンの助力を受ける。

【1263】2. 4. 10 アイソレーション (Isolation)

あるプロセスは色々な条件のもとに、例えばオペレーション "partAll" を用いてその全ての知合いから分かれたり、トリップを開始したり、または終了したりすることによって、アイソレートされる。

【1264】プロセスを "isolate (アイソレートともいう)" することは、他のプロセスのアーチファクトによって保持されているプロセスのアーチファクトへの各々のリファレンスをボイドし、他のプロセスのアーチファクトをプロセスのアーチファクトが保持していることの各々のリファレンスをボイドすることである。2つの例外がある。第1に、プロセスが占有するプレイスは、プロセスへのリファレンスを保持するが、プロセスが所有するオブジェクトへのリファレンスは保持しない。第2に、プロセスは、プロセスが占有するプレイスに対して参照し続けることができる。

【1265】2. 4. 11 終了 (Termination)

あるプロセスは、そのフレームがプロセスの実行スレッド中に含まれているメソッドを優雅に終了させることができる方法で終了される。

【1266】(i) A、B、Cがプロセスであり、(ii) AのアーチファクトがBのアーチファクトのフィーチャを要求し、(iii) BのアーチファクトがCのアーチファクトのフィーチャを要求し、(iv) Aがその固有のもしくはローカルなパーミットを生ずるものとする。その場合エンジンはAを次のようにして終了させる。

【1267】A、B、Cが同一性なければ、エンジンは "Permit Exhausted (パーミット消尽)" を投出し、例外のためのオペレーション "catch (キャッチ)" を用いて、最も最近に作り出されたA、BまたはCの方法に制御を戻す。方法は再開されるがAは、それに方法が関係しているオブジェクトであるかまたはそのオブジェクトのオーナーである、ローカルパーミットA、BまたはCのいずれかを保持している。メソッドが成功または失敗によって終了すると、エンジンはAの消尽されたパーミットを回復し、前述の終了アルゴリズムを再開する。

【1268】エンジンは、これをした後に、(i) Aをホールドし、(ii) Aをアイソレートし、(iii) 効力のある OAMポリシーがそのようにそれを必要とする

ならば診断的な検査のためにAを提供し、(iv) Aを破壊する。

【1269】2. 5 ネットワーク・モデル  
 インストラクションセットは、このセクションが定義する "network model (ネットワーク・モデル)" を実現する。

【1270】2. 5. 1 エージェント (Agents)  
 "agent (以下、エージェントともいう)" は、プレイスからプレイスへと移動しうるプロセスである。エージェントは、オペレーション "go" によって特別な目的点にそれ自身を搬送することができ、またオペレーション "send" によって同時に1つ以上のクローンにいくつかの目的点に向かって移動するように委託することができる。

【1271】注：インストラクションセットの "go" 及び "send" の各オペレーションは、インストラクションセットの最も顕著で有力な様相である。多くのネットワーク・アーキテクチャにおいて全てのプロセスは静止しており、メッセージの交換によって通信する。このネットワーク・アーキテクチャでは、あるプロセスは静止しているが他のプロセスは移動しており、前者は後者によって通信する。これはパラダイム・シフトである。

【1272】2. 5. 2 プレイス (Places)  
 "place (プレイス)" は、ゼロまたはそれ以上の他のプロセスのための場所であり、これらの他のプロセスの各々は、プレイスを参照し従ってプレイスと相互作用することができる。

【1273】注：プレイスは、インストラクションセットのただ1つの変革においてのアクションの例を表している。プレイスはその間を移動するエージェントによって相互作用する。

【1274】エンジン・プレイスと仮想的なプレイス (Engine Place vs Virtual Places)

2つの種類のプレイスがある。各々のエンジンは1つの "engine place (エンジンプレイス)" を保持しておりこれはエンジン自身を表している。また各々のエンジンはゼロまたは以上の "virtual (仮想的な)" プレイスも持つ。

【1275】サブプレイスとスーパープレイス (Subplace vs Superplace)

エンジンプレイスを除くいかなるプレイスも別のプレイスを占有することができる。あるプレイスが直接または間接に別のプレイスを占める場合、前者は後者の "subplace" であり、後者は前者の "superplace" である。あるプレイスが別のプレイスを直接占有する場合、前者は後者の "immediate subplace" であり、後者は前者の "immediate superplace" である。

【1276】プレイスハイアラーキ (Place Hierarchy)

エンジンが保持するプレイスは、ツリーのノード、すな

わち "place hierarchy" として表され、ここにノードはプレイスを表し、ノードの間のアークは、ノードが表すプレイスの間の占有の相互関係を表している。

【1277】注：ツリーのルートは、エンジンの場所を表し他のノードは仮想的なプレイスを表す。

【1278】2. 5. 3 トリップ (Trips)  
"トリップ (trip)" は、1つのプレイスすなわちトリップの "origine (始まりを意味し、オリジンともいう)" から同じプレイスの別の点、すなわちトリップの "destination (目的点)" へのエージェントの移動である。トリップは、そのオリジンのエージェントまたは "transit (トランジットともいう)" のプレイス内のエージェントを残すと、そのオリジンあるいはその目的点のいずれか一方だけ失敗させ得る。

【1279】注：あるトリップは、そのトリップが論理的なコミュニケーションのメディアでなく、物理的な手段のエージェントで伝送を行うことをときどき課すので、物長時間を要す。

【1280】オペレーション "go" とオペレーション "send"

エージェントは、オペレーション "go" によって単一の目的点にそれ自身移動するか、またはオペレーション "send" によっていくつかの目的点に同時に1つ以上のクローンが移動するように委託することができる。

【1281】注："send" オペレーションは、エンジンが保有するプレイスにいくつかのクローンが送られているにもかかわらず、ある特別なエンジンにエージェントの単一の代表を搬送する。このエンジンによれば、コンピュータのメモリスペースがさらに節減される。

【1282】制限 (Restrictions)  
オペレーション "go" または "send" は、クラス "Agent" のサブクラスに固有の方法のプロシージャかまたはこのプロシージャのアイテムであるプロシージャのみによって、反復的に要求される。エンジンはさもなければ、"State Improper (状態不適切)" を投出する。

【1283】2. 5. 4 チケット (Tickets)  
"チケット (ticket)" は、トリップをするエージェントの見地から見た、予定されたトリップを記述する。チケットの主な目的は、トリップの目的点を特定することである。

【1284】構成  
チケットは、目的点のテレネーム及びテレアドレスと、目的点がその1つのメンバであるクラスへのサイティションと、エージェントが目的点において必要とするローカルパーミットと、トリップの理想的な終了時間と、どんな状況のもとにおいてもトリップが成功または失敗となるべき時間とを包含する。

【1285】注：所望の時間を定義すると、移動中のエージェントの間にコミュニケーションリソースをどのように分配するかを定める上にネットワークの助けにな

る。

【1286】注：プログラミングの上の便利さのために許可される最大の時間を規定する効果は、オペレーション "制限" によっても達成される。

【1287】充足 (Satisfaction)  
チケットは、最大の時間内にエージェントが到達することに合致する、特定されたネーム、アドレス及びクラスのどんなプレイスによっても充たされる。

【1288】プレイスがチケットを充たさない場合、ネットワークはそのチケットを拒絶する。

【1289】2つ以上のプレイスのセットがチケットを充たさない場合、ネットワークは、1つのものがエージェントの到達に合致するかまたはネットワークが到達する全てのプレイスがエージェントの到達を拒絶するまで、セットの次々のメンバに向けてトリップを試みる。ネットワークがセットのメンバにアプローチする順序、ネットワークの、セットのカバレッジの程度が完全かどうか、並びに、チケットの提示後にセットに加わるプレイスにネットワークがアプローチするか否か、は定められていない。

【1290】2. 5. 5 ミーティング (Meetings)  
"meeting (以下、ミーティングともいう)" は、どちらもペティションされるオブジェクト及び従ってエージェントである同一のプレイスの2つの占有者が互いに相互作用する1つの機会である。要求エージェントは "petitioner (ペティショナ)" であり、レスポন্ディングエージェントは "petitionee (ペティショニ)" である。

【1291】ミーティング (Meeting)  
あるエージェントは、別のエージェントに、オペレーション "meet (ミート)" を介して、"会合 (meet)" することを求めることができる。エンジン-ペティショナではない-は、ペティショニのオペレーション "meeting" を求め、アーギュメントとして、ペティショナーのコンタクトを供給する。ミーティングはオペレーションの結果如何である。

【1292】オペレーション "meeting" の成功によってミーティングが完了した場合にのみそれら2つのエージェントに、相互に対する参照 (リファレンス) を与える。2つのエージェントは、ミーティングしている間、相互に対して "acquaintances (知人)" であると言われる。

【1293】エンジンはエージェントとのミーティングを系列化しない。エージェントがすでにオペレーション "meeting" を行なっているということは、エンジンがオペレーション "meeting" の遂行を再び要求することを妨げるものではない。エージェントは、例えばリソースによって、エージェントが求めるミーティングの系列化を与えることができる。

【1294】パーティング (Parting)

2つのエージェントのどちらか一方は、オペレーション"part"または"partAll"によってそのフレンドの一方または全てから"パート" (part: 分かれる) ことができる。エンジン-エージェントではないは、エージェントの1つ以上のフレンドのオペレーション"parting"を求めることができる。エンジンは、オペレーション"meeting"またはオペレーション"meeting"の結果のアーギュメントとしてエンジンが先に提供したコンタクトをアーギュメントとして提供する。ミーティングと異なって、成功するパーティングは、オペレーション"parting"の結果には依存しない。パーティングは直ちにそして無条件に生ずる。

【1295】エンジンは、各々のエージェントの、他のエージェントのアーチファクトへのリファレンスをポイドにする。

【1296】エンジンは、エージェントからのパーティングを系列化しない。エージェントがすでにオペレーション"parting"を遂行しているということは、エンジンがオペレーション"parting"の遂行を再び要求することを妨げない。エージェントは、例えばリソースによって、エージェントが求める系列化をパーティングによって与えることができる。

【1297】両方のエージェントが相互から分かれることを要求する場合、エンジンがどちらかのオペレーション"parting"を求めるか否かについては定められていない。

【1298】制限 (Restrictions)  
オペレーション"meet"、"part"、または"partAll"は、クラス"Agent"のサブクラスに固有の方法のプロシージャまたはこのようなプロシージャの1つのアイテムであるプロシージャのどちらかによってのみ、回帰的に要求されるべきである。エンジンはさもなければ、"State Improper (状態不適切)"を投出する。

【1299】2. 5. 6 ペティション (Petitions)  
"ペティション (petition)"は、ペティシヨナの見地からの予定されたミーティングを記述する。その主な目的は、ペティシヨニを特定することである。

【1300】構成  
ペティションは、ペティシヨニのテレネーム、ペティシヨニがその1つのメンバであるクラスのサイティション、並びにどんな状態においてもミーティングの試みが成功するかまたは失敗するべき時間からなる。

【1301】注: プログラミングの便宜上便宜の問題として許容される最大時間の規定の効果も、オペレーション"restruct"によって達成され得る。

【1302】充足 (Satisfaction)  
あるペティションは、最大時間内にミーティングに同意する特定されたネーム及びクラスのペティションされたオブジェクトによって充たされる。ペティシヨニは、最大時間のある点において (必ずしもそのスタート点では

ない) ペティシヨナと同一のプレイスの占有者でなければならない。

【1303】いかなるペティションされるオブジェクトもペティションを充たさない場合、エンジンはそのペティションを拒絶する。

【1304】2つ以上のペティションされるオブジェクトの組がペティションを充たした場合、エンジンは、1つのものがミーティングに同意するかまたはエンジンがアプローチする全てのペティションされるオブジェクトがミーティングを拒絶するまで、セットの次々のメンバとのミーティングを試みる。エンジンがセットのメンバにアプローチする順序、エンジンの、セットのカバレッジの程度がどの程度完全であるか、並びにペティションが提示された後にセットに参加するエージェントにエンジンがアプローチするか否かについては定められていない。

【1305】2. 5. 7 占有 (Occupation)  
あるプレイスの占有者達は、経時的に変化し得る。

【1306】エントリー  
あるプロセスは、2つの方法のどちらかであるプレイスに"エンタ (enter)"し、従って、占有することができる。あるエージェントは、トリップの結果としてそのプレイスに到着することができ、また、すでにその場所にある1つのプロセスは、その場所の別のプロセスを作り出すことができる。どちらの場合にも、エンジン-到着したエージェントまたは作り出したプロセスではない-その場所のオペレーション"entering"を要求する。到着したエージェントまたは作り出されたプロセスのためのコンタクトは、オペレーション"entering"のアーギュメントとして供給される。エントリーの成功は、オペレーションの結果にかかっている。

【1307】エンジンは、エントリーが完成した時のみ、すなわち、オペレーション"entering"の成功によってのみ、相互に対するリファレンスを、そのプレイスとそのプレイスにエンタしたプロセスとに与える。プロセスは、プレイスを占有している間、そのプレイスの"占有者 (occupant)"であるとされる。

【1308】エンジンは、あるプレイスへのエントリーを系列化しない。そのプレイスがオペレーション"entering"をすでに行なっているという事実は、オペレーション"entering"を再びエンジンが要求することを妨げない。そのプレイスは、プレイスが要求するいかなるエントリーの系列化も与えなければならない。

【1309】注: たとえトリップの出発点と目的点とが同一であったとしても、オペレーション"entering"は発生する。

【1310】エクシット  
あるプロセスは、次の2つの方法のどちらかであるプレイスを"exit (抜けを意味し、エクシットともいう)"し、従って、もはやその場所を占めないようになること

ができる。あるエージェントは、トリップの結果としてその場所を去ることができ、または、その場所にあるプロセスはそれ自身を破壊またはその場所にいる別のプロセスを破壊することができる。どちらの場合にもエンジンは一立ち去るエージェントでも破壊するプロセスでもない—その場所の、オペレーション”exiting”を求める。オペレーション”entering”のアーギュメントとして供給された同一のコンタクトは、オペレーション”exiting”のアーギュメントとして供給される。エクシットの成功は、エントリとは異なって、後者のオペレーションの成功に依存しない。エクシットは、すでに生じたものである。

【1311】エンジンは、プレイス及びそのプレイスをエクシットするプロセスの、その後のアーチファクトへのリファレンスをボイドとする。さらにエンジンは、そのプロセスをアイソレートする。

【1312】エンジンは、あるプレイスからのエクシットを系列化しない。そのプレイスがすでにオペレーション”exiting”を行なっているという事実は、オペレーション”exiting”を再びエンジンが要求することを妨げない。そのプレイスは、プレイスが要求するいかなるエクシットの系列化も供与しなければならない。

【1313】注：たとえトリップの出発点と目的点とが同一であってもオペレーション”exiting”は、成立する。

【1314】2. 5. 8 コンタクト (Contacts)  
エンジンは、プレイス及びエージェントに、それらのものの占有者及び知人との接触をそれぞれ保つ上の助力を与える。

【1315】占有者 (Occupants)  
プレイスがコンタクトされるオブジェクトでもある場合、エンジンは、属性がそのプレイスの現在の占有者を含み、そのプレイスの以前の占有者を含むことがないようにするためプレイスのコンタクトの属性を保持する。

【1316】エンジンは、プロセスがそのプレイスに入ったり出たりする時点においてプロセスに対するコンタクトをそのプレイスの属性に対してそれぞれ含めたり排除したりする。

【1317】知人 (Acquaintances)  
ペティションされるオブジェクトはコンタクトされるオブジェクトでもある場合、エンジンは、オブジェクトの現在のフレンドを常に含み、その以前のフレンドは含むことがないようにそのコンタクトの属性を保持する。

【1318】エンジンは、オペレーション”meet”または”meeting”が成功する時に、ペティションまたはペティションのためのコンタクトをそれぞれペティションの属性またはペティションの属性に含めるものとする。

【1319】エンジンは、どちらかのプロセスがオペレーション”part”または”partAll”を要求した場合に、ペティション及びペティションの属性からコンタ

クトを除去する。

【1320】2. 5. 9 サイテーション (Citations)

ネットワーク内において、あるクラスのオブジェクトはサイテーションによって表される。

【1321】サイテーション (Citations)

”サイテーション (Citation)”は、タイトルによって0またはそれ以上の引用されたオブジェクトを特定し、任意には、オーサー、エディション、またはオーサとエディションとの両方を特定する。サイテーションは、オーサーのオーソリティによってオーサを特定し、任意には、オーサのアイデンティティを特定する。もしあるサイテーションがこれらの3つの特徴の内のどれかを定義しないままに残した場合、サイテーションは、サイテーションが定義する特徴とともに、全ての引用されたオブジェクトを表すものとする。

【1322】オーサー

”オーサー (author)”は、引用されたオブジェクトを作り出すプロセスである。従って、オーサは、テレネームによって特定される。あるタイトルの全てのエディションのオーサは仲間である。

【1323】タイトル (Title)

”タイトル (title)”は、相互に対して後方向または前方向にコンパティブルであると主張された、引用された複数のオブジェクトの1組である。あるタイトルは、そのタイトルのオーサの共通のオーソリティに関連して解釈される識別子によって表される。

【1324】エディション (Edition)

”エディション (edition)”は、あるタイトルにおいての引用されたどれかのオブジェクトである。あるエディションは2つの整数によって表される。タイトルに対して相対的に解釈される一方の整数は、メジャーエディションを意味する。第1のものに対して相対的に解釈される他の整数は、マイナーエディションを表す。

【1325】注：あるタイトルの第1のメジャーエディションまたはマイナーエディションには、通常、数1または0がそれぞれ割り当てられる。

【1326】割り当てられるサイテーション

”割り当てられるサイテーション (assigned citation)”は、オーサー、タイトル及びエディションによって、引用されたオブジェクトを表し、オーサーを、そのオーソリティ及びそのアイデンティティの両方によって特定する。従って割り当てられたサイテーションは、引用されたオブジェクトを、他の全ての引用されたオブジェクトから、従って割り当てられたサイテーション自身のタイトル中の他のものから識別する。以下のテレネームについての説明参照。

【1327】コンパティビリティ

ある引用されたオブジェクトO<sub>2</sub>は、別の引用されたオブジェクトO<sub>1</sub>に対して次の場合にのみ”後の方向にコ

ンパティブル (backward compatible) ” である。その場合とは、両方の引用されたオブジェクトをナンバとするあるクラスによって規定された種類の変化のみを $O_1$ に対して行なうことによって $O_2$ が生成されることができ、その変更の種類が、プロセス $O_1$ について書かれたプログラムによって $O_2$ も同様に処理されることが確保されるように選定される場合である。同様の条件のもとに、 $O_1$ は $O_2$ に対して”前方向にコンパティブル”である。

【1328】2. 5. 10 テレネーム (Telenames)  
ネットワーク内においてある複数のクラスのオブジェクトは、テレネームによって表される。

【1329】テレネーム (Telename)

”テレネーム (telename)” は、テレネームが不適切である場合に0を、またはそれ以上の、名前のあるオブジェクトを、それらのもののオーソリティによって特定し、任意には、それらのもののアイデンティティによって特定する。テレネームがそのアイデンティティを定義されないままに残しておく場合、テレネームは、特定されたオーソリティーの全ての名前のあるオブジェクトを意味する。

【1330】オーソリティ (Authority)

ある名前のあるオブジェクトの”オーソリティ (authority)” は、そのオブジェクトに対して責任のあるエンティティ、例えば人または組織である。オーソリティは、ネットワーク中の他の全てのものからそのオーソリティを識別するオクテット文字列によって特定される。

【1331】2つのテレネームは、それらが同一のオーソリティを特定する場合にのみ”ピアス (仲間)” である。

【1332】注：オーソリティは、プログラムによってではなく政策的に作り出される。しかしアイデンティティは、両方の方法によって作り出される。

【1333】アイデンティティ (Identity)

名前のあるオブジェクトの”アイデンティティ (identity)” は、名前のあるオブジェクト自身である。アイデンティティは、オーソリティと同様に、オクテット文字列によって特定され、このオクテット文字列はそれだけでもネットワーク中の他の全てのものからそのアイデンティティを識別する。

【1334】注：あるオーソリティまたはアイデンティティを表すオクテット文字列は、人には意味を持たないことがある。インストラクションセットは、オクテット文字列がどのようにして割り当てられるかを定義しない。

【1335】割り当てられるテレネーム (assigned Telenames)

”割り当てられるテレネーム” は、オーソリティ及びアイデンティティによって正確に1つの名前のあるオブジェクトを表す。

【1336】2. 5. 11 テレアドレス (Teleaddresses)

ネットワーク中においてプレイスはテレアドレスによってロケートされる。

【1337】”ネットワーク (network)” は、全てのエンジンプレイスを含み、1つ以上の領域に区画される。”領域 (region: 以下、領域という)” は、特別のオーソリティによって操作されるかまたは提供される1つ以上のエンジンプレイスである。

【1338】テレアドレス

”テレアドレス (teleaddress)” は、(テレアドレスが不適切であれば) ゼロを、または1つ以上のプレイスを、それらの領域によって、また任意にはそれらのロケーションによって特定する。テレアドレスがそれらのロケーションを定義されないままに残した場合、テレアドレスは、特定される領域内の全てのプレイスを表す。

【1339】領域 (Region)

あるプレイスの”領域” は、そのプレイスを含む領域である。ある領域は、オーソリティと同じ仕方では特定されない (テレネームについての前記の説明参照)。従って領域は、それをネットワーク中の他の全てのものから識別するオクテット文字列によっては特定されない。

【1340】ロケーション (Location)

あるプレイスの”ロケーション (location)” は、そのプレイスの領域のオーソリティによってその目的のために選定されたプレイスの任意の特徴である。あるロケーションは、そのロケーションを領域内の全ての他のものから識別する文字列によって特定される。

【1341】ある領域内の2つのプレイスは同一のロケーションを用い得る。

【1342】注：ある場所のロケーションは例えばそれを指示するコンピュータ・システムでも有り得る。

【1343】注：あるロケーションを表す文字列は、人に対して意味を持ちうるがそれは必ずしも必要ではない。インストラクションセットは、文字列がどのようにして割り当てられるかを定義せず、割り当ての仕方は一般に領域ごとに異なる。

【1344】ルーティングアドバイス (Routing Advice)

テレアドレスは、テレアドレスが表すプレイスへのエージェントのルーティングについてのアドバイスを与えることができるが、これは必ずしも必要ではない。テレアドレスは、エージェントがそれを通してルーティングされる1つ以上の領域を示唆することによってこのアドバイスを与える。これらの”ルーティングアドバイス

(routing advice)” がない場合、例えばエージェントのソースから非常に離れたプレイスについては、ネットワークは、エージェントをこれらのプレイスに送り届けることはできないかもしれない。

【1345】割り当てられるテレアドレス (Assigned T

eleaddress)

”割り当てられるテレアドレス (assigned teleaddresses)” は、1つ以上のプレイス—これはいくつかのプレイスが同一のロケーションを用いるためである—を、それらの各々の領域及び領域内のロケーションによって特定する。

#### 【1346】2. 5. 12 相互変換 (Interchange)

”相互変換されたオブジェクト (interchanged object)” は、相互変換されたオブジェクトがダイジェストを持つ場合にオブジェクトが随伴するべきトリップをオブジェクトのオーナーが行なう場合にはいつでも、例えば目的点において見出される同等のオブジェクトによって、ネットワークは代替することのできる不変のオブジェクトである (この代替は必ずしも不可欠ではない)。ダイジェストは、以下に説明されている。

【1347】注：相互変換されたオブジェクトは、例えば目的点で見出される同等のオブジェクトによって代替することによってエンジンが時に相互変換されたオブジェクトを物理的に搬送することを避けるようにすることによって、オペレーション”go”及び”send”の遂行を改良する。

【1348】注：クラス及びパッケージは相互変換されたオブジェクトである。

#### 【1349】ダイジェスト

相互変換されたオブジェクトは、そのダイジェストが相互変換されたオブジェクトのものと同等であるオブジェクトのクラスのいかなる他のインスタントとも同等であると見なされる。”ダイジェスト”は、この目的のために適した、ゼロ以外の任意のオブジェクトである。例えばダイジェストは、相互変換されたオブジェクトの標準的な2進表示の数学的なハッシュ (細切れ: hash) であり得る。

【1350】注：相互変換されたオブジェクトは、オブジェクトがそのダイジェストよりも大きい場合にのみ、オペレーション”go”及び”send”の遂行を改善する。

#### 【1351】2. 6 タイムキーピング・モデル (Time keeping Model)

インストラクションセットは、このセクションが定義する”タイムキーピングモデル (timekeeping model)”を実現する。

【1352】タイムキーピング (計時) は2つの方法でなしうる。時間またはカレンダー時間は日付及び時間を同じように特定するがこれら2つは他の点では異なっている。

【1353】注：時間はカレンダー時間に変換でき逆も真である。

【1354】注：エンジンはカレンダー時間よりも一層コンパクトに内部的に時間を表すことができる。従って時間は一般に日付及び時間を保存し搬送するのにはより適している。

#### 【1355】2. 6. 1 時間 (Time)

”時間 (time)” は、協定世界時 (UTC) を用いて1秒の精度で、日付及び1日の時間を特定する。また時間は、時間が計測された時間帯を特定し、また昼光を節約時間 (DST) が、もし使用されていればであるが、どの程度までそのプレイスでその時間に有効であったかを特定する。

【1356】注：UTCは、実際には、グリニッチ標準時間 (GMT) として以前知られていたものである。

#### 【1357】2. 6. 2 カレンダー時間 (Calendar Time)

”カレンダー時間 (calendar time)” は、時間の全ての特徴及び他のものを有する。特に、カレンダー時間は、時間と分と秒とを検査及び変更にかける。カレンダー時間は、グレゴリー暦、その年の月及びその月の日を表す。また、週の日及び年をも表す。また、時間帯のUTCからの恒久的なオフセットを分で表し、恒久的なものからの季節的なオフセット (分) も表す。時間は、DSTのオフセットがゼロでない場合にDSTである。

#### 【1358】ノーマリティ (Normality)

カレンダー時間は、通常ある定められた範囲内にある整数として、近接可能な日、及び日の時間の各面を作成する。時間は [0, 24] の範囲内でまた分は [0, 60] の範囲内でまた秒は [0, 60] の範囲内でそれぞれ表示される。秒は [0, 60] の範囲内でその内閏年を勘案するために61の可能な値によって表示される。1993年は例えば整数1993によって表され、月は [1, 12] の範囲内で表され、日は [1, 31] の範囲内で表示される。週の日は、それぞれ日曜、月曜、火曜、水曜、木曜、金曜及び土曜を表す数字1、2、3、4、5、6及び7によって表される。月の日は [1, 366] で表される。恒久的及び季節的なオフセットはどちらも [-720, 720] で表される。

【1359】注：整数は、A. D. 年を、また、負数は、B. C. 年をそれぞれ表す。

#### 【1360】アブノーマリティ (Abnormrity)

カレンダー時間は、その整数値がその通常の範囲外にある面を有する。カレンダー時間は、例えば9月32として日付を表すことがありこれは10月2日を意味する。このような異常な数値は、有用な仕方、例えば月の日に2を加えることによってカレンダー時間を操作することによって生じうるが、カレンダー時間が正確と考えられる前に除去しなければならない。カレンダー時間はこの意味では、ノーマライゼーションプロセスによって正常にされる。

#### 【1361】規格化 (Normalization)

カレンダー時間は、次のステップによって”ノーマライズ (規格化)” される。第1のステップは、カレンダー時間のどの面も0でありうることを考慮している。

#### 【1362】デフォルト (Defaults)

・ 時間、分または秒は、0であれば、0とされる。  
年、月または日は0であれば0とされる。恒久的または季節的なオフセットは、0であれば、現在の時間を表す正常化されたカレンダー時間のものとされる。

#### 【1363】オフセット (Offsets)

・ 恒久的及び季節的なオフセットは、オフセット24×60すなわち整数1440で割算することによって得た剰余を[−720, 720]に転送の結果によって置き換えられる。

#### 【1364】時間 (Time)

・ 秒は、60を繰り返し加算または減算する（それぞれ分に1を足したり引いたりすること）によって通常の範囲とされる。分は、60を繰り返し加算または減算（それぞれ時間に1を足したり引いたりすること）によって正常な範囲にされる。時間は、24を繰り返し加算または減算（各々1を日にそれぞれ減算または加算すること）によって正常な範囲にされる。

#### 【1365】日付 (Date)

・ 月は、12を繰り返し加算または減算する（それぞれ1を年に足したり引いたりすること）によって正常な範囲にされる。日は、Kを加算または減算（それぞれ月に1を引いたり足したりすること）によって、[1, K]に置かれる。Kは、問題の年の問題の月の日の数である。1の加算または減算によって日が所用の範囲に置かれない場合には、この全体のステップを再び行なう。換言すれば、月は、再び日を調整する前に前述したようにして調整される。

#### 【1366】日 (Days)

・ 週及び年の日は、正確にセットされる。

#### 【1367】ローカライズとグローバルイズ (Localize vs Globalize)

カレンダー時間、それが特定する絶対時点を変えずに別のパナメントまたは季節のオフセットを表すように修正することができる。カレンダー時間は、そのオフセットが現在のプレイスのものとされた場合、"ローカライズ (localize; すなわち、地域化)" される。カレンダー時間は、そのオフセットがUTCのものとされた場合には、"グローバルイズ (globalize; 全域化)" される。

#### 【1368】2. 7 パターンマッチングモデル (Pattern Matching Model)

インストラクションセットは、このセクションが定義する"パターンマッチングモデル (pattern matching model)" を実現する。

#### 【1369】2. 7. 1 パターン (Patterns)

"パターン (pattern)" は、文字列を辞書的に分析する手段である。それ自身文字列である、パターンの"テキスト (text)" は、ストリングがパターンにマッチングされる基準を規定する。

【1370】パターンのテキストは、下記のシンタック

ティックな規則及びそれに不随するセマンティックな規則に従う一連のトークンである。BNF (Backus-Naur またはBackus Normal Form) で表される規則

は、("[" と "]" ) によって任意のトークンを囲む。これらの規則は、一方では任意のパターンのテキストを記述する上のBNFに従った使用を、他方ではある特別なテキストのメタキャラクタとしての可能な使用から識別するために、クォーテーションマーク (' ') によって、垂直バー ("|")、左ブラケット ("[" ) 及び右ブラケット ("]" ) を囲む。

【1371】あるパターンのテキストの各々のトークンはゼロまたはそれ以上のキャラクタである。テキストは、トークンとこのようなキャラクタとを連結することによって得られる。

#### 【1372】2. 7. 2 構造 (Structure)

##### Pattern

パターン、すなわちそのテキストは、この規則に従う。:

pattern ::= オールタナティブ["|"pattern]

ストリング (文字列) は、それがその他のオールタナティブに一致している場合にのみパターンに一致させる。

#### 【1373】オールタナティブ (Alternative)

オールタナティブは、次の規則に従う。:

オールタナティブ ::= ["|"]コンポーネント{ \$ }

コンポーネント ::= Component [Components]

ストリングは、間にギャップを置かない、連続するサブストリングが、オールタナティブの次々のコンポーネントとマッチする場合にのみオールタナティブにマッチする。オールタナティブが、カレット ("^") で始まるかまたはドルマーク ("\$\$") で終わる場合、サブストリングの系列は、ストリングの最初及び終わりにおいてそれぞれ開始し、そして終了しなければならない。

#### 【1374】コンポーネント (Component)

コンポーネントは、次の規則に従う。:

コンポーネント ::= Item[\* | + | ?]

あるストリング (文字列) は、0 またはそれ以上のサブストリングを含み各々が、アスタリスク ("\*") が表れる場合にアイテムにマッチする) を含むかまたはプラス・マーク ("+") が表れるならば1つ以上のこのようなサブストリングを含む場合にのみあるコンポーネントにマッチする。どちらもクエッション・マーク ("?") が表れるならばアイテムにマッチするかまたはクリアされたまたはさもなければアイテムにマッチする。

#### 【1375】アイテム (Item)

アイテムは、次の規則に従う。:

コンポーネント ::= (Pattern) |  
"["CharacterClass"]" |  
Character

あるストリング (文字列) は、パターン、キャラクタクラス (Character Class) またはキャラクタの内アイ



テムによって規定されるどれかにマッチする場合にのみアイテムにマッチする。

#### 【1376】CharacterClass

キャラクタクラスは次の規則に従う。：

CharacterClass ::= [ ] CharacterItems

ストリングは、キャレット ( ` ) が存在している場合にのみ、キャラクタアイテム (Character Items) のリストに整合する場合にのみ、キャラクタクラスにマッチする。

#### 【1377】CharacterItems

キャラクタ・アイテムのリストは次の規則に従う。：

CharacterItems ::= CharacterItem [CharacterItems]

CharacterItem ::= CharacterRange | Character

文字列は、文字列の連続するサブストリングがその間にギャップなしに、リスト中の次々のキャラクタアイテムとマッチする場合にのみ、キャラクタアイテムのリストにマッチする。サブストリングの系列は、ストリングの開始点で開始され、終了点で終了しなければならない。

#### 【1378】CharacterRange

キャラクタレンジは次の規則に従う。：

CharacterRange ::= Character - Character

ストリング (文字列) は、それが第1のキャラクタの1つ後の文字かまたは第1キャラクタを含み、第2のキャラクタの1つ前または第2のキャラクタに等しい1つのキャラクタを持つ場合にのみキャラクタレンジにマッチする。

#### 【1379】キャラクタ (Character)

キャラクタは次の規則に従う。：

Character ::= . | \ metacharacter | character

ストリングが1つの文字を含み、ピリオッド ( ` . ` ) が存在する場合、にのみキャラクタにマッチし、逆スラッシュ ( ` \ ` ) が存在する場合にメタキャラクタでありその他の場合はキャラクタである。

【1380】注：従って逆スラッシュ ( ` \ ` ) がメタキャラクタの直前にある場合には、メタキャラクタの特別の意味は避けられる。

#### 【1381】2. 7. 3 他の非ターミナル

Character

メタキャラクタ以外のクラス "Character" のインスタンス。

#### 【1382】メタキャラクタ (Metacharacter)

メタキャラクタ (metacharacter)

#### 2. 7. 4 メタキャラクタ (Metacharacters)

パターンのテキストは、メタキャラクタを含める。メタキャラクタは、テキストを支配するセマンティックルールに含まれる特別な意味を有する。メタキャラクタは、表A. 5に示されたキャラクタ並びに、キャラクタのテレスクリプトの "ストリング (String) " トークンに示されたもの例えばバックスペースである。

#### 【1383】

#### 【表5】

表A. 5	
メタキャラクタ	意 味
垂直バー (   )	2者の分離
脱字記号 ( ` )	ストリングの始まりにアンカーする (anchor) かマッチングするか規則を反転
ドル記号 ( \$ )	ストリングの終わりにアンカーする (anchor)
左丸括弧 ( ` ( ` )	入れ子 (nested) パターンの始まり
右丸括弧 ( ` ) ` )	入れ子 (nested) パターンの終わり
左角括弧 ( ` [ ` )	キャラクタクラスの始まり
右角括弧 ( ` ] ` )	キャラクタクラスの終わり
アスタリスク ( * )	アイテム0回またはそれ以上
プラス記号 ( + )	アイテム1回またはそれ以上
クエッションマーク ( ? )	任意な項目
マイナス記号 ( - )	範囲を定義するキャラクタを分離
句点 ( . )	あるキャラクタとの一致
逆スラッシュ ( \ )	以下のメタキャラクタの除去

### 3. テレスクリプトクラスの概観

オブジェクト指向されたインストラクションセットは、このアペンディックスのこのセクションにおいて概観されるある予め規定されたクラスを含む。これらのクラスは次のように複数のグループに分けられる。サブセクションは各々のグループに向けられる。各々のサブセクション内において1つのサブセクションはグループ内の各々のクラスに向けられる。

【1384】解釈されるインストラクションセットは、

例えば制御フローのためのステートメント形式を規定するのではなく、そのクラスの実行に依存する。従ってインストラクションセットのセマンティックスは予め規定されたクラスのセマンティックスであることが多い。

#### 【1385】3. 1 グループ (Groups)

インストラクションセットの予め規定されたクラスは、その1つがカーネルと呼ばれる複数のグループに分けられる。この区画は、教育的な目的のみに用いられ、カー

ネル以外のグループにおけるクラスがカーネルにより実行され得ることを示さず、また、エンジンがカーネルのクラスを実行することがあるが他のグループのものは必ずしも実行しなくてもよいことを示すものでもない。

【1386】次のグループが規定されている。

【1387】カーネル (Kernel)

このグループは、基本機械言語、例えばクラス、オブジェクト、リファレンス、プロシージャ、実行及び例外を規定する。

【1388】プリミティブ (Primitives)

このグループは、不可分の情報、例えばブーリアン、オクテット、数、キャラクタ及び時間の基本的な形式を与える。

【1389】コレクション (Collections)

このグループは、複合情報、例えばセット、リスト、スタック、ストリング及び辞書の基本的な形式を与える。

【1390】クラス定義 (Class Definition)

このグループは、クラス定義例えばインタフェース、属性、オペレーション、実施及び方法の基本的な部分を与える。

【1391】特定化 (Identification:アイデンティフィケーション)

このグループは、ネーミング及びアドレッシング、例えばテレネーム及びテレアドレスを与える。

【1392】プロセス (Processes)

このグループは、マルチ・タスク例えばプロセス、パーミット、リソース及びコンタクトを与える。

【1393】エージェント及びブレイス (Agents and Place)

このグループは、ネットワーク内のプロセスの移動、例えばプロセス、エージェント及びチケットを与える。

【1394】ミーティング (Meeting)

このグループは、ネットワークノード、例えばミーティングノード及びペティション内のプロセス相互作用を与える。

【1395】その他 (Miscellaneous)

このグループは、いろいろなこと例えばランダムナンバジェネレータ及びパターンマッチャを与える。

【1396】これらのグループは、以上に示した順序で以下に示される。おのおののグループ内のクラスはアルファベット順で示される。

【1397】3. 2 カーネルグループ (Kernel Group)

オブジェクト (リファレンスされる)

- ・ クラス (サイトされそして相互変換される)
- ・ コレクション
- ・ セット (検査される)
- ・ コンストレインド・セット (コンストレインド)
- ・ パッケージ (引用され相互変換される)
- ・ コンストレイント

- ・ 例外 (変更されない)
- ・ プログラミングの例外
- ・ カーネルの例外
- ・ 実行の例外
- ・ 期待されない例外
- ・ プリミティブ (実行されそして変換されない)
- ・ 識別子 (順序付けされる)
- ・ 有資格の識別子
- ・ マーク
- ・ モディファイヤ
- ・ ゼロ
- ・ プロシージャ
- ・ セレクタ

コンストレインド

実行される

リファレンスされる

変換されない

検査される

3. 2. 1 クラス (Class)

オペレーション

convert

isInstance

isMember

isSubclass

new

”Class” は、オブジェクトのセット、クラスのインスタンス、を定義する不変のオブジェクトである。

【1398】クラスの固有のオペレーションは、オペレーション”new”を介して新しいインスタンスを作り出し、別のクラスのインスタンスを、このクラスのインスタンスにオペレーション”convert”によって変更し、あるオブジェクトはオペレーション”isInstance”によるインスタンスまたはクラスのオペレーション”isMember”を介したメンバであるか否かを表示し、別のクラスがオペレーション”isSubclass”を介して現在のクラスのサブクラスであるか否かを表示する。

【1399】3. 2. 2 コンストレインド (Constrained)

属性

Constraint

”constrained object (コンストレインドオブジェクト)” は、オブジェクトがその1つのメンバである別のクラスによって特定されたそのプロパティのものに対して制約を課す。

【1400】コンストレインド・オブジェクトの固有の属性はコンストレイント (属性”constraint”) である。

【1401】注：コンストレインドディクショナリ、リスト及びセットは、コンストレインドオブジェクトである。

### 【1402】3. 2. 3 コンストレイント (Constraint)

#### 属性

classId  
isInstance  
isOptional  
ofClass  
passage

” constraint” は、他のオブジェクトに課せられた制限を規定するオブジェクトである。

【1403】コンストレイントの本来の属性は、コンストレイントのサブジェクトがそのメンバとなるべきクラスの識別子（属性” classId”）、クラス自身（属性” ofClass”）、サブジェクトがそのクラスのインスタンスとなるか否か（属性” isInstance”）、サブジェクトがゼロであるか否か（属性” isOptional”）及びサブジェクトの通路（属性” passage”）である。

### 【1404】3. 2. 4 例外 (Exception)

オペレーション

throw

” 例外 (exception)” は、遂行または実行の失敗を記述するオブジェクトである。

【1405】例外の本来のオペレーションは、例外を投出する（オペレーション” throw”）。

### 【1406】3. 2. 5 実行される (Executed)

オペレーション

catch

do

either

if

loop

repeat

while

” 実行されるオブジェクト (executed object)” は、プロシージャの1つのアイテムとして許可される。

【1407】実行されるオブジェクトの本来のオペレーションは、(i) オペレーション” do” を介して1回、(i i) オペレーション” catch” を介してある例外を1度キャッチする。(i i i) オペレーション” if” を介して予条件が充たされた場合にのみ1度、(i v) オペレーション” リピート” を介して数回、(v) オペレーション” loop” を介して不特定数の何回か、(v i) オペレーション” while” を介して別の実行されたオブジェクトの遂行が指示する回数がある。別のオペレーションは、2つの実行されるオブジェクトの内どちらを実行すべきかを定める（オペレーション” either”）。

【1408】注：クラス” Primitive” の予め規定された具体的なサブクラス並びにクラス” Constrained List” の予め規定されたサブクラスのインスタンスは、実行されるオブジェクトである。

### 【1409】3. 2. 6 実行の例外

” 実行の例外 (execution exception)” は、エンジンがあるプロシージャのアイテムを実行しえないことを示すカーネル例外である。

【1410】このクラスの予め規定されたサブクラスの中には、インストラクションセットのあるアスペクトをエンジンがインプリメントし得ないことをそのメンバの1つが示す” 内部例外 (Internal Exception)” がある。理想的なエンジンは、このような例外を投出しませんが、実際のエンジンは、それを行なう。後者のドキュメンテーションによれば、例えば2つの整数の和がエンジンによって内部的に便利に表示されるには、大きすぎる場合を挙げている。

### 【1411】3. 2. 7 識別子 (Identifier)

” 識別子 (idenntifier)” は、特定のコンテキストにおいてあるクラスを別のクラスから識別するプリミティブである。

### 【1412】3. 2. 8 カーネル例外 (Kernel Exception)

” カーネル例外 (Kernel Exception)” は、このセクションのクラスのメンバによって投出されたプログラミングの例外である。

### 【1413】3. 2. 9 マーク (Mark)

” マーク (mark)” は、1つの可能な値、” マーク (mark)” についてプリミティブである。あるマークは、一連のオブジェクト通常はあるスタックの最も情報のアイテムを区別するために用いられる。

### 【1414】3. 2. 10 モディファイヤ (Modifier)

” モディファイヤ (moifier)” は、以下の可能な値” demarcate”、” getClass”、” getProperty”、” getVariable”、” mention”、” setAttribute”、” setProperty”、” setVariable” 及び” useStack” についてプリミティブである。あるプロシージャにおいてモディファイヤの直後にあるアイテムの実行を変えるために用いられる。

### 【1415】3. 2. 11 ゼロ (Nil)

” ゼロ (nil)” は、1つの可能な値、” ニル (nil)” についてプリミティブである。ゼロは例えば別のクラスのあるメンバがスタック上に存在しないことを示すために用いられる。

### 【1416】3. 2. 12 オブジェクト (Object)

属性

class

size

オペレーション

copy

finalize

initialize

isEqual

select  
内部オペレーション  
getAttribute  
getClass  
getProperty  
getVariable  
setAttribute  
setProperty  
setVariable

ある”オブジェクト”は、情報及び情報処理のインストラクションセットのユニットである。あるオブジェクトはあるクラスのインスタンスである。

【1417】オブジェクトの本来の属性は、そのクラス(”class”)及びその大きさ(”size”)である。

【1418】オブジェクトの本来のオペレーションは、オブジェクトの初期化(オペレーション”initialize”)及び最終化(オペレーション”finalize”)し、オブジェクトが別のオブジェクトに等しいかどうかを定め(オペレーション”isEqual”)いくつかのオブジェクトの内では以前のオブジェクトが等しいものはどれかを定め、そのオブジェクトと対をなす実行されるオブジェクトを遂行し(オペレーション”select”)、別のしかしそれ自身の同一のコピーを生成する(オペレーション”copy”)。

【1419】あるオブジェクトは、いくつかの内部的フィーチャを有する。”内部的(internal)”フィーチャは、クラス”Object”によってシールされ、さらに、クラス”Object”のユーザによって定義されたサブクラスの固有のフィーチャが同一の特定子(identifier)を持つことを妨げないシステムフィーチャである。従って、内部的フィーチャは、虚構的なもの(fictitious)である。その唯一の目的は、インストラクションセット特にその実行モデルの定義を簡単にすることである。

【1420】あるオブジェクトの内部的オペレーションは、オブジェクトのクラスのインタフェース及びインプリメンテーションにおいて使用された特定子を作動時間の間オブジェクトに拘束する。従ってオブジェクトは、

(i) ある識別子が意味するそれ自身の属性をオペレーション”getAttribute”を介して取得しうるかまたはオペレーション”setAttribute”を介してセットし、(ii) ある識別子が意味するそれ自身のプロパティをオペレーション”getProperty”を介して取得し、またはオペレーション”setProperty”を介してセットし、(iii) ある識別子が意味する変数をオペレーション”getVariable”を介して取得し、またはオペレーション”setVariable”を介してセットし、並びに(iv) オペレーション”getClass”を介してある識別子が意味するクラスを特定しロケートする。

【1421】3. 2. 13 パッケージ(Package)  
”パッケージ(package)”は、拘束されたクラスのセ

ットである。さらに、パッケージは、パッケージが含む各々のクラスの仲間である引用されたオブジェクトである。

【1422】1つのパッケージは、次の種類の変更のみを行なうことによって後のものから先のものが作られた場合にのみ、別のパッケージに対して後方向にコンパティブルである。第1に、新しいクラスが付加されうる。第2に現存のクラスは、その現存のクラスと後方向にコンパティブルな別のものに変えることができる。

【1423】3. 2. 14 プロシージャ(Procedure)

”プロシージャ(procedure)”は、実行されるオブジェクトのある順序付けられたセットを含むプリミティブなものである。

【1424】3. 2. 15 プログラミングの例外(Programming Exception)

”プログラミングの例外(programming exception)”は、あるフィーチャの用意または使用においてプログラミングエラーが生じたことを示す例外(exception)である。

【1425】3. 2. 16 有資格の識別子(Qualified Identifier)

”有資格の識別子”は、フィーチャの実施がそれに対して固有かまたは受け継がれた関係にあるフィーチャまたはクラスを表す識別子である。

【1426】3. 2. 17 リファレンス(Referenced)

属性

isProtected

オペレーション

discard

isSame

protect

ref

”参照される(referenced)”オブジェクトは、それに対するリファレンスを取得し得るオブジェクトである。そのフィーチャは、それを要求するために用いられた、オブジェクト自身ではない、リファレンスを操作する。

【1427】リファレンスされるオブジェクトの固有の属性は、あるリファレンスがプロテクトされているかを否か明らかにする(属性”isProtected”)。

【1428】リファレンスされるオブジェクトの固有のオペレーションは、所望ならばプロテクト(オペレーション”protect”)されたオブジェクトへの新しいリファレンス(オペレーション”ref”)を生成し、現存のリファレンスを廃棄し(オペレーション”discard”)、2つのリファレンスが同一のオブジェクトのものであるかどうかを明示する(オペレーション”isSame”)。

【1429】注：全てのオブジェクトは、リファレンス

されるオブジェクトである。その場合現在のクラスは純粋に教育的な理由によって定義される。

【1430】3. 2. 18 セレクタ (Selector)  
 "セレクタ (selector)" は "break"、"client"、"continue"、"escalate"、"here"、"process"、"self" 及び "succeed" の可能な値についてプリミティブである。あるセレクタは、特別の実行効果を得るために用いられる。

【1431】3. 2. 19 変更されない (Unchanged)  
 "変更されない (unchanged)" オブジェクトは、変化させることができない。"変化 (change)" の意味は、変化しないオブジェクトが1つのメンバである他のクラスに依存する。変更されないオブジェクトは、しばしば "不変 (immutable)" と記述される。

【1432】注: "クラス"、"例外"、"パッケージ"、"プリミティブ" 及び "時間" のクラスのメンバは、不変である。

【1433】3. 2. 20 予想されない例外 (Unexpected Exception)

属性

exception

"予想されない例外 (unexpected exception)" は、あるフィーチャが、そのフィーチャが明らかにしないクラスの例外を送出したことを示す実行例外処理である。

【1434】予想されない例外の固有な属性は、明らかにされない例外 (属性 "exception") である。

【1435】3. 2. 21 検査 (Verified)

オペレーション

verify

"検査されるオブジェクト (verified object)" は、そのオブジェクトが1つのメンバである他のクラスに依存する1つ以上の仕方でも内部的に不一致とすることができる。

【1436】検査されるオブジェクトの本来のオペレーションは、そのオブジェクトが内部的に一致するか否かを示す (オペレーション "verify") 。

【1437】注: セットは、検査されるオブジェクトである。

【1438】3. 3 プリミティブグループ (Primitive Group)

オブジェクト (リファレンスされる)

- ・ 例外 (変更されない)
- ・ プログラミングの例外
- ・ プリミティブな例外
- ・ プリミティブ (実行され変更されない)
- ・ ビット (順序あり)
- ・ ブーリアン (順序あり)
- ・ キャラクタ (ケースドアンドオーダード)
- ・ 数 (順序あり)

・ 整数

・ 実数

・ オクテット (順序あり)

・ テレナンバ

・ 時間 (順序あり、変更なし)

ケースド (cased)

順序あり (Ordered)

3. 3. 1 ビット (Bit)

"ビット (bit)" は、2つの可能な値、すなわち "0" と "1" とを有するプリミティブなものである。

【1439】3. 3. 2 ブーリアン

オペレーション

and

not

or

"ブーリアン (boolean)" は、2つの可能な値、"偽" と "真" とを有するプリミティブなものである。

【1440】ブーリアンの固有のオペレーションは、論理的な肯定 (オペレーション "and")、択一 (オペレーション "or") 及び否定 (オペレーション "not") を可能にする。

【1441】3. 3. 3 ケースド (Cased)

属性

isLower

isUpper

オペレーション

makeLower

makeUpper

"ケースド (Cased)" オブジェクトは、キャラクタの情報を含む。この情報やこのようなケースドオブジェクト自身は、アップパーケース (大文字)、ロウアーケース (小文字) またはその混合にすることができる。

【1442】ケースドオブジェクトの固有の属性は、あるキャラクタ情報が小文字であるか (属性 "isLower") または大文字であるか (属性 "isUpper") を表す。

【1443】ケースドオブジェクトの固有の操作は、オブジェクトのすべてを小文字 (オペレーション "makeLower") またはすべて大文字 (オペレーション "makeUpper") に等価なものを作り出す。

【1444】注: 2つのケースドオブジェクトは、オペレーション "makeLower" またはオペレーション "makeUpper" と組合せてオペレーション "isEqual" を使用することによって、そのケースと関わりなく比較することができる。

【1445】注: キャラクタとストリングはケースド・オブジェクトである。

【1446】3. 3. 4 キャラクタ (Character)

"キャラクタ (character)" は、その可能な値がユニコードキャラクタ [ユニコード] を基本とする。

## 【1447】3. 3. 5 整数 (Integer)

オペレーション

modulus

quotient

”integer”は、整数でもある数である。

【1448】整数の固有のオペレーションは、整数の除算を行ない、商（オペレーション”quotient”）または結果としての剰余（オペレーション”modulus”）のいずれか一方を生ずる。

【1449】注：内部的に、エンジンは、有限な確度のみを持って整数を表すことができる。従って、全部の整数が、実際に、このクラスのインスタンスであるとは限らない。

## 【1450】3. 3. 6 数 (Number)

オペレーション

abs

add

ceiling

divide

floor

multiply

negate

round

subtract

truncate

”数 (number)”は、基本的な演算操作を可能にする基本的なものである。

【1451】ある数の固有の操作は、加算（オペレーション”add”）、減算（オペレーション”subtract”）、掛算（オペレーション”multiply”）、除算（オペレーション”divide”）、否定（オペレーション”negate”）、絶対値（オペレーション”abs”）、丸め（オペレーション”round”）、切捨て（オペレーション”truncate”）及びフロア（オペレーション”floor”）及びシーリング（オペレーション”ceiling”）の各機能を遂行する。

## 【1452】3. 3. 7 オクテット (Octet)

”オクテット (octet)”は、8ビットから成る基本的なものである。リファレンスの目的のためにビットは、ビット7ービット0によって示される。

## 【1453】3. 3. 8 順序あり (Ordered)

オペレーション

isAfter

isBefore

max

min

”順序あり (Ordered)”のオブジェクトは、任意の1つのものを他のものの前に、後に、またはそれと同じ所に置くことによってそのメンバが順序付けされた別のクラスの1つのメンバにする。”前”及び”後”の意味

は、そのクラスに依存する。

【1454】順序付けされたオブジェクトの固有の操作は、1つのかかるオブジェクトが別のものの前に（オペレーション”isBefore”）、または別のものの後に（オペレーション”isAfter”）あるかそして2つの内のどちらが最初（オペレーション”min”）または最大（オペレーション”max”）であることを示す。

【1455】注：2つの順序付けされたオブジェクトが一致しているか否かは、オペレーション”isEqual”で示される。

【1456】注：多くのクラスのインスタンスは、順序付けされたオブジェクトである。

【1457】3. 3. 9 プリミティブ (Primitive)  
”プリミティブ (primitive)”は、インストラクションセットの基本的なオブジェクトである。

【1458】注：各々のインストラクションセット・プリミティブがこのクラスのメンバであるわけではない。

【1459】3. 3. 10 プリミティブの例外 (Primitive Exception)

”プリミティブの例外 (primitive exception)”は、このセクションのクラスのメンバによって送出されたプログラミングの例外である。

【1460】3. 3. 11 テレナンバ (Telnumber)  
属性

country

extension

telephone

”テレナンバ (telnumber)”は、国際電話ネットワークにおいて、電話機のアドレス、すなわち電話番号を与えるオブジェクトである。

【1461】テレナンバの固有の属性は、電話機が置かれている国のコード（”country”）、その国が電話機に与えるナンバ（”telephone”）及びもし電話機に内線があればその電話機の内線（”extension”）のコードである。

## 【1462】3. 3. 12 時間 (Time)

オペレーション

adjust

interval

”時間 (time)”は、UTCを用いて1秒の精度まで1日の日付及び時間を特定するオブジェクトである。また、時間は、それが計測された時間帯を記録し、また、もし何かあれば、その時その場所のDSTの実際の範囲も記録する。

【1463】時間の固有のオペレーションは、任意の数の秒によって時間を調節し（オペレーション”adjust”）その時間と別の時間との間の時間間隔（オペレーション”interval”）を与える。

【1464】3. 4 コレクショングループ (Collecti

on Group)

オブジェクト (リファレンスされる)

- ・ アソシエーション (順序あり)
- ・ コレクション
  - ・ ・ リスト (順序あり)
  - ・ ・ ・ コンストレインドリスト (コンストレインド)
  - ・ ・ ・ ・ ビットストリング (実行される)
  - ・ ・ ・ ・ オクテットストリング (実行される)
  - ・ ・ ・ ・ ストリング (ケースドアンドエグゼキューテッド)
  - ・ ・ ・ スタック
  - ・ ・ セット (検査される)
  - ・ ・ ・ コンストレインドセット (コンストレインド)
  - ・ ・ ・ ディクショナリ
  - ・ ・ ・ ・ コンストリンドディクショナリ (コンストレインド)
  - ・ ・ ・ ・ ・ レキシコン
- ・ 例外 (変更なし)
- ・ ・ プログラミングの例外
- ・ ・ ・ コレクションの例外
- ・ ストリーム

ハッシュド (Hashed)

### 3. 4. 1 アソシエーション (Association)

属性

key

value

”アソシエーション (association)” は、あるオブジェクトの ”key” 及び ”value” である、2つのオブジェクトを含む1つのオブジェクトである。

【1465】アソシエーションの固有の属性は、キー (属性 ”key”) 及びその値 (属性 ”value”) である。

【1466】注：アソシエーションは、ディクショナリを作り出すために用いられる。

【1467】3. 4. 2 ビットストリング (Bit String)

”ビットストリング” は、クラス ”ビット” のインスタンスをアイテムとするコンストレインドリストである。

【1468】3. 4. 3 コレクション (Collection)

属性

length

オペレーション

clear

examine

exclude

include

stream

”コレクション (collection)” は、0またはそれ以上の他のオブジェクト、そのプロパティに含まれるその ”items” の順序付けされないセットを含むオブジェクトである。コレクションの ”長さ (length)” は、コレ

クションのアイテムの数である。コレクションは、その長さが0の時に ”クリア (cleared)” される。クラス ”collection” のサブクラスは、そのメンバのアイテムを制限することができる。

【1469】コレクションの固有の属性はコレクションの長さ (属性 ”length”) である。

【1470】コレクションの固有のオペレーションは、新しいアイテムとしてのオブジェクトを含め、(オペレーション ”include”)、現在のアイテムを排除し (オペレーション ”exclude”)、全ての現在のアイテムを排除し (オペレーション ”clear”)、アイテムの検査の準備をし (オペレーション ”examine”)、コレクションのアイテムをアイテムとするストリーム (オペレーション ”stream”) を生成させる。

【1471】注：コレクションの長さは、無制限である。

【1472】3. 4. 4 コレクションの例外 (Collection Exception)

”コレクションの例外 (collection exception)” は、このセクションのクラスのメンバによって投出されたプログラミングの例外である。

【1473】3. 4. 5 コンストレインドディクショナリ (Constrained Dictionary)

”コンストレインドディクショナリ (constrained dictionary)” は、条件付けによってそのキーが制限されたディクショナリであり、単一の制限は、全体の辞書を支配する。

【1474】3. 4. 6 コンストレインドリスト (Constrained List)

”コンストレインドリスト (constrained list)” は、条件 (constraint) によってアイテムが制限されたりリストであり、単一の条件は、全体のリストを支配する。

【1475】3. 4. 7 コンストレインドセット (Constrained Set)

コンストレインドセットは、条件によってアイテムが制限されたセットであり、単一の条件は、全体のセットを支配する。

【1476】3. 4. 8 ディクショナリ (Dictionary)

オペレーション

add

drop

find

get

rekey

set

transpose

”ディクショナリ (dictionary)” は、組合せ (association) をアイテム (items) とするセットである。クラス ”dictionary” のサブクラスは、そのメンバのキー、



値または両方を制限することができる。

【1477】ディクショナリの固有の操作は、あるキーと新しいアイテムとしての別のある値との間の組合せを含め（オペレーション”add”）、あるキーとの組合せを排除し（オペレーション”drop”）、あるキーの組み合わせられた値を定め（オペレーション”get”）、あるキーの組み合わせられた値を取り替え（オペレーション”set”）、ある値の組み合わせられたキーを定め（オペレーション”find”）、あるキーを別のキーと取り替え（オペレーション”rekey”）及び2つのキーを転置する（オペレーション”transpose”）。

【1478】3. 4. 9 ハッシュド (Hashed)

属性

”Hashed”オブジェクトは、ハッシュファンクションがそれについて定義されている別のクラスの一つのメンバである。そのクラスの各々のメンバは、それ自身のハッシュ、整数、を計算し、2つのメンバは同一であればそれらのもののハッシュは、同一であることを確実にする。

【1479】ハッシュドオブジェクトの固有の機能は、そのオブジェクトについてもハッシュファンクション（属性”hash”）である。

【1480】注：ディクショナリのキーがハッシュドオブジェクトであれば、ディクショナリのオペレーションのいくつかのものの遂行において属性”hash”を使用することができるが、これは必ずしも必要ではない。

【1481】注：予め定義されたクラスのいかなるインスタンスもハッシュドオブジェクトではない。

【1482】3. 4. 10 レキシコン (Lexicon)

”レキシコン (lexicon)”は、キーが識別子であるコンストレインドディクショナリである。

【1483】3. 4. 11 リスト (List)

オペレーション

add  
drop  
find  
get  
reposition  
set  
transpose

”リスト (list)”は、そのアイテムが [1, n] で番号付けされるコレクションである。”n”はコレクションの長さである。アイテムの長さは、リスト中のそのアイテムの”位置”である。

【1484】あるリストの固有のオペレーションは、新しいアイテムとしてもオブジェクトを含め（オペレーション”add”）、現在のアイテムを排除し（オペレーション”drop”）、あるアイテムの検査の取決めをし（オペレーション”get”）、現在のアイテムを置き換え（オペレーション”set”）、現在のアイテムの位置を

定め（オペレーション”find”）、現在のアイテムの再位置決めし（オペレーション”reposition”）、2つの現在のアイテムを転置する（オペレーション”transpose”）。

【1485】3. 4. 12 オクテットストリング (Octet String)

”オクテットストリング (octet string)”は、そのアイテムがクラス”Octet”のインスタンスであるコンストレインドリストである。

【1486】3. 4. 13 セット (Set)

オペレーション

difference  
intersection  
union

”セット (set)”は、含められた時にそのアイテムのどの2つの同一ではないコレクションである。

【1487】あるセットの固有の操作は、2つのセットのユニオン（オペレーション”union”）または区分（オペレーション”intersection”）を計算し、他のものにおいて他のもののアイテムに等しいあるもののアイテムを排除する（オペレーション”difference”）。

【1488】注：あるアイテムのそのままの変更は、2つのアイテムを同一のものにのこしておくことがある。

【1489】3. 4. 14 スタック (Stack)

オペレーション

pop  
push  
pushItems  
roll  
swap

”スタック (stack)”は、アイテムが付加された順序と逆の順序において通常そのアイテムがドロップされるリストである。スタックの”トップ (top)”は、位置1であり、スタックの”下部 (bottom)”は、スタックの長さである位置である。

【1490】スタックの固有の操作は、上部にオブジェクトを付け加え（オペレーション”push”）、リストのアイテムを上部に付け加え（オペレーション”pushItems”）、上部のアイテムをドロップし（オペレーション”pop”）、上部の2つのアイテムを転置し（オペレーション”swap”）及びもっとも上方のアイテムのある位置数だけローリングする（オペレーション”roll”）。

【1491】3. 4. 15 ストリーム (Stream)

属性  
current  
isDone  
next

”ストリーム (stream)”は、組み合わせられたオブジェクトの系列すなわちストリームの”アイテム (item

s) ” 中のオブジェクトに順次アクセスするためのオブジェクトである。ストリームは同一のアイテムを2回”生成 (produces) ” させることはなく、最後のアイテムを生成させた後は0を生じる。

【1492】ストリームの固有の属性は、要求に応じて次のアイテムをつくり出し (属性” next” )、もっとも最近生成させたアイテムを再生し (属性” currnt” ) 及びその最後のアイテムをまだ生成させているか否かを定める (属性” isDone” ) を可能にする。

【1493】注：ストリームのあるアイテムそれ自身が0である場合、この事実は、” ネクスト (next) ” または” カレント (current) ” 属性の解釈に当たって考慮されなければならない。

【1494】3. 4. 16 ストリング (String)

オペレーション

substring

” ストリング (string: 文字列) ” は、クラス” character ” のインスタンスをアイテムとするコンストレインドリストである。

【1495】ストリングの固有のオペレーションは、サブストリングのコピーをつくり出すことである (オペレーション” substring” )。サブストリングは、あるストリング中の連続する位置を示す0または1以上の文字の系列である。サブストリングは、2つの位置  $P_1$  及び  $P_2$  によって定義され、サブストリングのアイテムは、ストリングの位置  $[P_1, P_2]$  であるアイテムである。

【1496】3. 5. クラス定義グループ

オブジェクト (参照される)

- ・ クラス定義
- ・ 例外 (変更なし)
- ・ ・ プログラミングの例外
- ・ ・ ・ クラスの例外
- ・ フィーチャ
- ・ ・ 属性
- ・ ・ オペレーション
- ・ インプリメンテーション
- ・ インターフェース
- ・ 方法

3. 5. 1 属性 (Attribute)

属性

constraint

isSet

” 属性定義 (attribute definition) ”、すなわちクラス” 属性 (Attribute) ” のメンバは、属性のためのフィーチャの定義である。

【1497】属性定義の固有の属性は、属性がどのようにして拘束されるか (属性” constraint” ) 並びにそれを設定できるか (属性” isSet” ) を定義する。前者は、オペレーション”  $\alpha$ Get” の結果に、そして属性が

セット可能であれば、オペレーション”  $\alpha$ Set” のアークギュメントに影響する。

【1498】3. 5. 2 クラス定義 (Class Definition)

属性

implementation

interface

majorEdition

minorEdition

title

オペレーション

makeClasses

” クラス定義 ” は、あるクラスを定義するオブジェクトである。

【1499】クラス定義の固有の属性は、クラスのインターフェース (属性” interface” )、実施 (属性” implementation” )、タイトル (属性” title” )、メジャーエディション (属性” majorEdition” ) 及びマイナーエディション (属性” minorEdition” ) を規定する。

【1500】クラスの定義の固有の操作は、1つまたは1つ以上のクラス定義のセットが定義する同じ数のクラスのセットをつくり出す (オペレーション” makeClasses” )。クラスの間の必要とされるリファレンスは、クラスが作成された時に設定される。オペレーションを要求するプロセスは、そのクラスの多さである。

【1501】3. 5. 3 クラスの例外 (Class Exception)

” クラスの例外 (class exception) ” は、このセクションのクラスのメンバによって投出されたプログラミンの例外である。

【1502】3. 5. 4 フィーチャ (Feature)

属性

exceptions

isPublic

” フィーチャ定義 ”、すなわちクラス” feature ” のメンバは、そのインターフェースに充分であるクラスの各々のメンバの特別なフィーチャを規定するオブジェクトである。

【1503】フィーチャの定義の固有の属性は、そのフィーチャがパブリックであるか否か、プライベートでないか (属性” isPublic” )、フィーチャの例外がメンバであるクラス (属性” exceptions” ) を規定する。システムフィーチャは、フィーチャ定義を受けない。

【1504】3. 5. 5 インプリメンテーション (Implementation)

属性

classMethods

fromMethods

instanceMethods

properties

setMethods

superclasses

toMethods

vocabulary

”インプリメンテーション (implementation)” は、インプリメンテーションがその一部分であるクラスに固有のまたはそれによって引き継がれた、選択されたフィーチャ及び変換を実施するオブジェクトである。

【1505】インプリメンテーションの固有の属性は、直接インプリメンテーションスーパークラス (属性”superClasses”) を特定し、選択されたクラスフィーチャ (属性”classMethods”)、インスタンスフィーチャ (属性”instanceMethods” 及び属性”setMethods”) 及び (属性”fromMethods”) 及び (属性”toMethods”) の変更を提供し、ユーザによって定義されたクラスを表すために用いられるインプリメンテーションを特定する (属性”vocabulary”)。

【1506】3. 5. 6 インターフェース (Interface)

属性

classFeatures

instanceFeatures

isAbstract

sealedClassFeatures

sealedInstanceFeatures

superclasses

vocabulary

”インターフェース (interface)” は、インターフェースがその一部分であるクラスに固有のフィーチャを定義するオブジェクトである。

【1507】インターフェースの固有の属性は、クラスが抽象的であるかを定め (属性”isAbstract”)、クラスのイミディエイトインターフェーススーパークラス (属性”superclasses”) を特定し、クラスの固有のクラスフィーチャ (属性”classFeatures”) 及びインスタンスのフィーチャ (属性”instanceFeatures”) を定義し、クラスフィーチャ (属性”sealedClassFeatures”) 及びそのクラスのシールするインスタンスフィーチャ (属性”sealedInstanceFeatures”) を特定し、ユーザによって定義されたクラスを表す上にインターフェースが使用する識別子を定義する (属性”vocabulary”)。

【1508】3. 5. 7 メソッド (Method)

属性

procedure

variables

”メソッド (method)” は、あるフィーチャをインプリメントするために使用されるオブジェクトである。

【1509】あるメソッドの固有の属性は、その方法のプロシージャ (属性”procedure”) 及びその方法の変

数 (属性”variables”) の識別子である。

【1510】3. 5. 8 オペレーション (Operation)

属性

arguments

result

”オペレーション定義 (operation definition)”、すなわちクラス”Operation”のメンバは、あるオペレーションのフィーチャの定義である。

【1511】あるオペレーションの定義の固有の属性は、オペレーションのアーギュメントが数について一定であるかまたは可変であるかを定め、後者の場合には、いかにして各々のアーギュメントが拘束 (コンストレイント) されるかを定める (属性”arguments”)。また属性は、オペレーションの結果を有するか否かを定め、結果を有する場合にはいかにその結果が拘束されるかを定める (属性”result”)。

【1512】3. 6 特定化グループ (Identification Group)

オブジェクト (リファレンスされる)

- ・サイテーション (順序有り)

- ・テレアドレス

- ・テレネーム

引用 (Cited)

名前付け (Named)

3. 6. 1 サイテーション (Citation)

属性

author

majorEdition

minorEdition

title

”サイテーション (citation)” は、テレスク립トネットワークにおいて、0 またはそれ以上のエディション、例えばクラスを特定するオブジェクトである。

【1513】サイテーションの固有の属性は、エディションがその中に現れるタイトルを規定し、(属性”title”)、任意に、そのタイトルのオーサー (属性”author”)、任意には、エディションのメジャーエディション (属性”majorEdition”)、マイナーエディション (属性”minorEdition”) または両方を定義する。

【1514】3. 6. 2 引用 (Cited)

属性

Citation

”引用される (cited)” オブジェクトは、割り当てられたサイテーションを持つオブジェクトである。2つの引用されたオブジェクトは、それらの割り当てられたサイテーションにおいて割り当てられたテレネームが仲間である場合にのみ仲間である。

【1515】引用されたオブジェクトの固有の属性は、そのオブジェクトの割り当てられたサイテーションであ

る（属性” citation”）。

【1516】注：クラス及びパッケージは、サイテッドオブジェクトである。

【1517】3. 6. 3 名前付け (Named)

属性

Name

”名前付けされた (Named)” オブジェクトは、割り当てられたテレネームを有するオブジェクトである。2つの名前付けされたオブジェクトは、それらのものの割り当てられたテレネームが仲間である場合にのみ仲間である。

【1518】名前付けされたオブジェクトの固有の属性は、そのオブジェクトの割り当てられたテレネーム（属性” name”）である。

【1519】注：プロセスは、名前付けされたオブジェクトである。

【1520】3. 6. 4 テレアドレス (Teleaddresses)

属性

location

provider

routingAdvice

”テレアドレス (teleaddress)” は、テレスクリプトネットワークに0または1つ以上のプレイスを配置するオブジェクトである。

【1521】テレアドレスの固有の属性は、そのプレイスの領域を与え（属性” provider”）、任意には、そのプレイスの位置（属性” location”）を与え、さらにルーティングアドバイス（属性” routingAdvice”）を与える。

【1522】3. 6. 5 テレネーム (Telename)

属性

authority

identity

”テレネーム (telename)” は、ネットワーク中において0または0以上のネーム付けされたオブジェクトを特定するオブジェクトである。

【1523】テレネームの固有の属性は、名前付けされたオブジェクトのオーソリティを規定し（属性” authority”）、任意に、単一に名前付けされたオブジェクトのアイデンティティを定める（属性” identity”）。

【1524】3. 7 プロセスグループ (Process Group)

オブジェクト（リファレンスされる）

- ・ コンタクト
- ・ 例外（変更なし）
- ・ ・ プログラミングの例外
- ・ ・ ・ プロセスの例外
- ・ パーミット（順序付けされる）
- ・ プロセス（名前付けされる）

・ リソース

コンタクティッド (Contacted)

3. 7. 1 コンタクト (Contact)

属性

subject

subjectClass

subjectName

subjectNotes

”コンタクト (contact)” は、あるプロセスについて他のものとの相互作用を表しドキュメントするオブジェクトである。

【1525】コンタクトの固有の属性は、サブジェクト（属性” subject”）、サブジェクトの割り当てられたテレネーム（属性” subjectName”）、サブジェクトのクラスの割り当てられたサイテーション（属性” subjectClass”）、ならびにオブザーバが保持するオブジェクト（属性” subjectNotes”）である。サブジェクト属性が0であっても、エンジンは他の属性の真正さを保証する。

【1526】注：典型的には属性” subjectNotes” は、オブザーバの見地から2つのプロセスの間の相互作用の状態である。

【1527】3. 7. 2 コンタクティッド (Contacted)

属性

contacts

必然的にプロセスである、”コンタクトされた (contacted)” オブジェクトは、そのオブジェクトのコンタクトのセットを保つ上にエンジンからの助力を受ける。

【1528】コンタクトされたオブジェクトの固有の属性は、オブジェクトのコンタクト（属性” contacts”）である。

【1529】注：いかなる予め定義されたクラスのインスタンスもコンタクトされるオブジェクトではない。

【1530】3. 7. 3 パーミット (Permit)

”パーミット (permit)” は、あるプロセスに能力を与えるオブジェクトである。

【1531】属性

age

authenticity

canCharge

canCreate

canDeny

canGo

canGrant

canRestart

canSend

charges

extent

priority

パーミットの固有の属性は、パーミットのサブジェクトの最大のエイジ（属性”age”）、サイズ（属性”extent”）、チャージズ（属性”charges”）、プライオリティ（属性”priority”）、及びオーセンティシティ（属性”authenticity”）を規定する整数である。

【1532】パーミットの固有の属性は、オペレーション”charge”（属性”canCharge”）、オペレーション”go”（属性”canGo”）、オペレーション”send”（属性”canSend”）またはオペレーション”new”を、クラス”Process”のサブクラスについてサブジェクトについて要求することをサブジェクトに許容し、あるプロセスのあるパーミットを経て能力を拒絶（属性”canDeny”）または許容（属性”canGrant”）し、さらに再スタート（属性”canStart”）をサブジェクトが要求することを許容する。

【1533】オペレーション

intersection

パーミットの固有の操作は、2つのパーミットのインターセクションをリターンする（オペレーション”intersection”）。

【1534】3. 7. 4 プロセスの例外（Process Exception）

”プロセスの例外（process exception）”は、このセクションのクラスのメンバによって投出されるプログラミングの例外である。

【1535】3. 7. 5 プロセス（Process）

属性

age

brand

charges

localPermit

permit

priority

privateClasses

regionalPermit

オペレーション

charge

live

restrict

restricted

sponsor

wait

”プロセス”は、自律的な計算を構成する名前付けされたオブジェクトである。

【1536】プロセスの固有の属性は、プロセスのエイジ（属性”age”）、ブランド（属性”brand”）、チャージズ（属性”charges”）、ローカルパーミット（属性”localPermit”）、パーミット（属性”permit”）、プライオリティ（属性”priority”）、私的に保有され、使用されるクラス（属性”privateClass

es”）並びに量的なパーミット（属性”regionalPermit”）である。

【1537】あるプロセスの固有のオペレーションは、そのプロセスの自律的な計算（オペレーション”live”）を行い、ある一定の期間の間計算を遅延させ（オペレーション”wait”）、プロセス自身に一時的なパーミットを課し（オペレーション”restrict”）、一時的なパーミット及びレスポンドのオーナーの結果としての実効的なパーミットをもとにプロシーダを遂行し（オペレーション”sponsor”）、現在のプロセスが他のプロセスに与えるサービスについて他のプロセスにチャージする（オペレーション”charge”）。

【1538】3. 7. 6 リソース（Resource）

属性

condition

conditions

オペレーション

use

”リソース（resource）”は、ある特定のプロシーダをプロセスが遂行する間プロセスに対してある保障を与えることができるオブジェクトである。

【1539】リソースの固有の属性は、リソースの現在の状態（属性”condition”）及びリソースの可能な状態（属性”conditions”）の識別子である。

【1540】リソースの固有のオペレーションは、そのリソースについて要求された保障がリソースによってなされた時にあるプロシーダを遂行する（オペレーション”use”）。

【1541】3. 8 エージェント及びプレイスグループ（Agent and Place Group）

オブジェクト（リファレンスされる）

- ・ オーセンティケータ
- ・ 例外（変更なし）
- ・ トリップの例外
- ・ 手段
- ・ プロセス（ネーミングされている）
- ・ エージェント
- ・ プレイス（移動しない）
- ・ チケットスタブ
- ・ チケット
- ・ ウェイ

変更なし（Unchanged）

- ・ 相互変換される

移動なし（Unmoved）

3. 8. 1 エージェント（Agent）

オペレーション

go

send

”エージェント”は1つのプレイスから次のプレイスに移動することのできるプロセスである。エージェントの

本来のオペレーションは、エージェントを1つの目的点に（オペレーション”go”）または1以上のクローンを同数の目的点に同時に（オペレーション”send”）搬送する。

【1542】3. 8. 2 オーセンティケータ (Authenticator)

”オーセンティケータ (authenticator)” は、例えばエージェントが領域の間を通過する間に、トリップを開始したエージェントをネットワークがそれによって認証することのできるオブジェクトである。

【1543】注：クラス”Authenticator”の具体的なサブクラスは、領域特異であり、したがってユーザによって定義されている。

【1544】3. 8. 3 相互変換 (Interchanged)

属性

digest

”相互変換された (digested)” オブジェクトは、オブジェクトのオーナーがトリップする時にネットワークが同等のオブジェクトによって変えることのできる変更されないオブジェクトである。

【1545】相互変換されたオブジェクトの固有の属性は、オブジェクトのダイジェストである（属性”digest”）。

【1546】3. 8. 4 手段 (Means)

”ミーンズ (means)” は、ネットワークがトリップを開始したエージェントをトリップのソースからトリップの目的点にまたはその両方に搬送することができる手段たとえば特別の無線通信メディアムを特定するオブジェクトである。

【1547】注：クラス”Means”の具体的なサブクラスは、領域特異であり、したがってユーザによって定義されている。

【1548】3. 8. 5 プレイス (Place)

属性

address

publicClasses

オペレーション

entering

exiting

terminate

”プレイス (place)” は、0またはより多くの他のプロセスのための場所であるプロセスである。

【1549】あるプレイスの固有の属性は、プレイスのアドレス（属性”address”）及びそのプレイスの公共に保持され使用されるクラス（属性”publicClasses”）である。

【1550】あるプレイスの固有のオペレーションは、リクエストが十分に能力を持ち（リクエストのパーミットの属性”canTerminate”）であり、しかも占有者の仲間であることを条件として、プロセスの提案されたプレイ

スのエントリに判断を下し（オペレーション”entering”）、プロセスがプレイスからエクシットしたことを記録し（オペレーション”exiting”）ならびに占有者を強制的に終了させる（オペレーション”terminate”）。

【1551】3. 8. 6 チケット (Ticket)

属性

desiredWait

destinationAddress

destinationClass

destinationName

destinationPermit

maximumWait

”チケット (ticket)” は、トリップを行うエージェントの見地から見た予定されたトリップを記述するチケットスタグ（半券のこと）である。チケットの主な目的は、トリップの目的点を特定することである。

【1552】チケットの固有の属性は、目的点のテレネーム（属性”destinationName”）及びテレアドレス（属性”destinationaddress”）、目的点がその1つのメンバであるクラスへのサイティション（属性”destinationClass”）、エージェントが目的点において必要とするローカルパーミット（属性”destinationPermit”）、理想的にトリップがその範囲内で行われるべき時間（属性”desiredWait”）、並びにいかなる状態においても、トリップが成功または失敗するべき事情（属性”maximumWait”）である。

【1553】3. 8. 7 チケットスタブ (TicketStub)

属性

travelNotes

way

”チケットスタブ (ticketStub)” は、トリップの結果を記述するオブジェクトである。

【1554】チケットのスタブの固有の属性は、トリップの出発点へのもどり道（属性”way”）ならびにトリップを開始したエージェントが専用的に使用する、チケット上に最初にのせられていた情報（属性”travelNotes”）である。

【1555】3. 8. 8 トリップの例外 (Trip Exception)

属性

ticketStub

”トリップの例外 (ticket exception)” は、あるチケットが記述するトリップを与える上にネットワークが失敗したことを表す例外である。

【1556】トリップの例外の固有の属性は、チケットスタブである（属性”ticketStub”）。

【1557】3. 8. 9 移動しない (Unmoved)

”unmoved” オブジェクトは、1つのプレイスから次の

プレイスに移動することはできない。移動しないオブジェクトはしばしば移動不可能と記述される。移動しないオブジェクトへのリファレンスは、そのオブジェクトの移動が試みられる場合に、ボイドされる。

【1558】注：プレイスは、移動不可能である。

【1559】3. 8. 10 ウェイ (Way)

属性

authenticator

means

name

”ウェイ (way)” は、トリップを開始したエージェントをネットワークがそれに沿って移送させることができるルート (道筋) を特定するオブジェクトである。

【1560】ウェイの固有の属性は、トリップがそれを経て行われるべき領域の名前 (属性”name”) 及びその領域に入るために用いられる手段 (属性”means”) 及びオーセンティケータ (属性”authenticator”) である。

【1561】3. 9 ミーティンググループ  
オブジェクト (リファレンスされる)

- ・ 例外 (変更なし)
- ・ ・ ミーティングの例外
- ・ ペティション
- ・ プロセス (名前付けされている)
- ・ ・ プレイス (移動しない)
- ・ ・ ・ ミーティングプレイス

ペティションド

3. 9. 1 ミーティングの例外 (Meeting Exception)

”ミーティングの例外 (meeting exception)” は、あるペティションが記述するミーティングをミーティングプレイスが取り計らうことができなかったことを意味する例外である。

【1562】3. 9. 2 ミーティングプレイス (Meeting Place)

オペレーション

meet

part

partAll

”ミーティングプレイス (meeting place)” は、その占有者の中に含まれる2つのペティションされるオブジェクトの間のミーティングを取り決めるためのプレイスである。ミーティングプレイスの固有のオペレーションは、ミーティングを開始させ (オペレーション”meet”)、それを終了させ (オペレーション”part”) ならびにリクエストを含むすべてのミーティングを終了させる (オペレーション”partAll”)。

【1563】3. 9. 3 ペティション (Petition)

属性

agentClass

agentName

maximumWait

”ペティション (petition)” は、ペティショナの見地から見た予定されたミーティングを記述するオブジェクトである。ペティションの主な目的はペティションの特定化である。

【1564】ペティションの固有の属性は、ペティションのテレネーム (属性”agentName”)、ペティションがその1つのメンバであるクラスへのサイティション (属性”agentClass”)、並びにミーティングの試みがどんな事情のもとにおいても成功または失敗するべき時間 (属性”maximumWait”) である。

【1565】3. 9. 4 ペティションド (Petitioned)

オペレーション

meeting

parting

必ずエージェントである、”ペティションド (petitioned)” オブジェクトは、ミーティングの設立においてペティショナまたはペティションの役目をする事ができる。

【1566】ペティションされたオブジェクトの固有のオペレーションは、ミーティングの要求について判断を下し (オペレーション”meeting”)、パーティングについて記録することである (オペレーション”parting”)。

【1567】注：定義されていないクラスのインスタンスは、ペティションドオブジェクトである。

【1568】3. 10 その他のグループ  
オブジェクト (リファレンスされる)

- ・ カレンダー時間
- ・ 例外 (変更なし)
- ・ ・ プログラミングの例外
- ・ ・ ・ その他の例外
- ・ パターン (順序有り)
- ・ プリミティブ (実行され変更されない)
- ・ ・ ナンバ (順序有り)
- ・ ・ ・ 実数
- ・ ストリーム
- ・ ・ ランダムストリーム

3. 10. 1 カレンダー時間 (Calendar Time)

属性

day

dayOfWeek

dayOfYear

dst

hour

minute

month

second

year

zone

オペレーション

globalize

localize

normalize

” カレンダ時間 (calendar time) は、UTCを用いて1秒の精度でローカルデイトと時間とを特定するオブジェクトである。カレンダ時間はまたローカル時間帯とDSTがその時その場所では有効となる度合いも記憶する。

【1569】カレンダ時間の固有の属性は、時間 (属性”hour”)、分 (属性”minute”)、秒 (属性”second”)、日付の年 (属性”year”)、月 (属性”month”)、日 (属性”day”)、週の日 (属性”dayOfWeek”)、年 (属性”dayOfYear”) ならびに最後に、時間帯 (属性”zone”) ならびにDSTがそこでどの程度有効か (属性”dst”) を与える。

【1570】カレンダ時間の固有のオペレーションは、カレンダ時間のローカライズ (オペレーション”localize”)、グローバルイズ (オペレーション”globalize”)、またはノーマライズ (オペレーション”normalize”) である。

【1571】3. 10. 2 種々の例外

” ミセレネアスエクスセプション (miscellaneous exception) ” は、このセクションのクラスのメンバによって投出されたプログラミングの例外である。

【1572】3. 10. 3 パターン (Pattern)

オペレーション

find

substitute

” パターン (pattern) ” は、ストリングをレキシコルによってアナライズするためのオブジェクトである。それ自身ストリングである、パターンのテキストは、分析の正確な性質を定める。

【1573】パターンの固有のオペレーションは、最初のマッチング、サブストリングのためのストリングをサーチし (オペレーション”find”) または全てのマッチングサブストリングを変更 (オペレーション”substitute”) する。

【1574】3. 10. 4 ランダムストリーム (Random Stream)

そのアイテムが疑似ランダムに選ばれたインテジャ

[0, M] であるストリームである。各アイテムは同じ間隔中の整数シーダーによって定められる。ランダムなストリームは、その最後のアイテムを生成させない。この発明の一事例によれば、” M ” は、ランダムストリームの創出者によって選択され、イニシャライゼーションパラメータとして特定される。

【1575】3. 10. 5 実数 (Real)

実数は、数学において実数を形成する数である。

【1576】注：内部的にエンジンは実数を表現し、限られた有限な確度で—実数を含む算術演算を行う。従ってすべての実数が必然的にこのクラスのインスタンスであるわけではなく、実数によって行われた算術演算の結果は、単に近似的であることがある。

【1577】4 テレスクリプトクラスの詳細

テレスクリプトのあらかじめ規定されたクラスは、アペンディックスのこのセクションにおいてアルファベット順に規定されている。サブセクションは各々のクラスに向けられる。

【1578】4. 1 コンベンション (Conventions)

クラスの定義は、そのクラスを包含するクラスグラフの部分の図によって開始される。クラス自身はアンダーラインされている。

【1579】以下のセクションは、各々のあらかじめ定義されたクラスの説明において、必要な限り示されている。

【1580】クラス

このセクションは、クラスのフィーチャとは別に、クラスを記述する。クラスは以前はセクション6のコンベンションに従って規定されているが、クラスのフィーチャの以前の定義のかわりに、省略 (...) がなされている。クラスのインターフェースの符号化がパラメータ化されていればそのパラメータがどのようになされているかがクローズ (散文) によって記述されている (下記のセクション6参照)。

【1581】構成

このセクションは、クラスのメンバに所属するオペレーション”initialize”を記述する。このセクションが省略されている場合、以下にセクション6に示すように特別な説明が内包されている。

【1582】サブクラス

このセクションは、クラスのあらかじめ規定された直接サブクラスの定義を含む。このセクションは、クラスがクラス”エクスセプション”のサブクラスでない場合のみ、または、クラス”Exception”のサブクラスであるがあらかじめ規定されたサブクラスをもたない場合のみ省略される。

【1583】属性

このセクションは、クラスに固有の属性を定義する。クラス自身及びそのクラスのインスタンスのパブリックな、プライベートな及び内部的な及びその他のシステムの属性に別々なセクションが向けられている。ひとつのセクションは、それが空であると思われる場合にのみ省略される。

【1584】属性は、セクション6のコンベンションに従って公式に規定されている。属性のセマンティクス及び例外は、非公式に散文によって定義されている。

【1585】オペレーション

以下のセクションは、ここに示された他のセクションが



クラスに固有に属性を定義するのと同じ仕方で、クラスに固有のオペレーションを定義する。

【1586】あるオペレーションが、そのアーギュメント及び結果を保持するスタックを変更する場合、オペレーションのスタックに対する影響は、アーギュメントがスタックからポップされ、いかなる結果もいまだスタックに保存されていないものと想定して記述される。

#### 【1587】アダプテーションズ

このセクションは、クラスが受け継いだフィーチャをクラスがどのようにしてインプリメントするかを記述する。このセクションは、クラスがそのようなフィーチャをインプリメントしない場合にのみ省略される。

#### 【1588】変換

このセクションは、クラスのインスタンスを生じさせる変換に記述する。このセクションは、そのような変換がない場合にのみ省略される。

【1589】個別の変換の定義は、ソースクラスを特定し、ソースクラスのメンバがどのようにしてサブジェクトクラスに変換されるかを記述する。すべてのメンバがこのように変換できない場合にはどれが変換され得るかが表示される。ひとつの変更のソースクラスが別の変換のソースクラスのサブクラスである場合には、以前の変換は後の変換に優先する。変換は鎖状になっていない。

【1590】注：変換は1ステップで行われる。従って、ストリングへのテレナンバの変換及びオクテットストリングの変換が規定されていないとしても、オクテットストリングへのテレナンバの変換は定義されていない。

#### 【1591】4. 2 エージェント (Agent)

オブジェクト (リファレンスされている)

- ・ プロセス (名前付けされている)
- ・ ・ エージェント

クラス

Agent:

抽象的なインターフェース (プロセス) = (...);

プライベートインスタンスオペレーション

go:

プロテクトされていないop (チケット: copied ticket);

TicketStubは、PermitViolated, StateImproper, TripExceptionを投出する。

【1592】チケットの他の条項に従ってチケットによって特定された目的点にレスポンドを搬送する (アーギュメント "ticket")。オペレーションは、チケットスタブをリターンする。

【1593】レスポンドはオペレーション "go" を禁止した場合 ("PermitViolated")、レスポンドのステイトがオペレーション "go" を排除した場合 ("StateImproper")、またはトリップが失敗した場合 ("TripException")。

#### 【1594】send:

保護されていないop (チケット: copied List (Ticket))

TicketStub | Nilは、

PermitViolated, StateImproper, TripExceptionを投出する。

【1595】チケットの他の条項に従って、ならびに各々のリストされたチケット (アーギュメント "tickets") によって特定された目的点にレスポンドのクローンを搬送する。レスポンドのアローワンスは、チケットがクローンに割り当てる量によって減少する。オペレーションは、レスポンドに0をリターンし、各々のクローンにチケットスタブをリターンする。

【1596】"PermitViolated" またはクラス "StateImproper" のどちらかのクラスのメンバがオペレーションを失敗に終わらせた場合、レスポンドは失敗を経験し、いかなるクローンもつくり出されていない。またレスポンドがオペレーションの成功を経験し各々のクローンが別々にそして独立して、クラス "TripException" のメンバであるためにオペレーションが成功しまたは失敗したことを経験する。

【1597】レスポンドがオペレーション "send" を禁止するパーミットを保持し、そのアラウワンスは、リストされたチケットのパーミットのもの合計よりも少ない場合 ("PermitViolated")、レスポンドのステイトは、オペレーション "send" を排除し、("StateImproper")、またはトリップは失敗する ("TripException")。

#### 【1598】4. 3 アソシエーション (Association)

オブジェクト (リファレンスされる)

- ・ アソシエーション (順序有り)

クラス

Association

シールされたインターフェース [keyClass, valueClass: Class]

(Object, Ordered) = (...);

サブジェクトクラスの各々のメンバの、キー (アーギュメント "keyClass") 及び値 (アーギュメント "ValueClass") がそれ自身メンバであるクラスによってパラメータ化される。

#### 【1599】構成

initialize

保護されていないop (key: keyClass; value: valueClass);

レスポンドの本来の属性を最後に述べたアーギュメントにする (アーギュメント "key" 及びアーギュメント "value")。

#### 【1600】パブリックインスタンスの属性

key:

keyClass  
レスポンドのキー  
value:  
valueClass;  
レスポンドの値。  
【1601】アダプテーション  
isAfter  
ひとつのアソシエーションは、それらのキーに従って次々に続く。  
【1602】isBefore  
あるアソシエーションは、それらのキーに従って次のものの前にある。  
【1603】isEqual  
2つのアソシエーションは、それらのもののキーに従って同等である。  
【1604】4. 4 属性  
オブジェクト (リファレンスされる)  
・ フィーチャ  
・ 属性  
クラス  
属性  
シールされたインターフェース (Feature) = (...);  
構成  
initialize  
保護されていないop (constraint:Constraint|Nil;  
isSet, isPublic: Boolean|Nil;  
exceptions: Set [Identifier!]|Nil);  
レスポンドの属性を、最後に述べたアーギュメント (アーギュメント "constraint", "exceptions", "isPublic" 及び "isSet") にする。アーギュメントが0である場合、対応する属性は、0をイニシャライゼーションのパラメータとしたコンストレイント、クリアされたセット "偽"、または、"偽"、とされる。  
【1605】パブリックインスタンス属性  
constraint:  
constraint;  
レスポンドが定義するアトリビュートに対するコンストレイント。  
【1606】isSet:  
Boolean;  
レスポンドが定義する属性がリードオンリでないかどうか。  
【1607】アダプテーションズ  
isEqual  
2つの属性の定義がフィーチャの定義、及び属性の定義に固有の属性に従って同等である。  
【1608】4. 5 オーセンティケイタ (Authenticator)  
オブジェクト (リファレンスされる)  
・ オーセンティケイタ

クラス  
Authenticator:  
抽象的なインターフェース = 0;  
4. 6 ビット (Bit)  
オブジェクト (リファレンスされる)  
・ プリミティブ (実行され、変更されない)  
・ ビット (順序有り)  
クラス  
Bit:  
シールされたインターフェース (Primitive, Ordered)  
= 0;  
アダプテーション  
isAfter  
第1のものが1であり第2のものが0である場合にのみあるビットはあるビットが次のビットのあとに続く。  
【1609】isBefore  
最初のビットが0で次のビットが1である場合のみひとつのビットが別のビットの前にある。  
【1610】isEqual  
2つのビットは、両方が0であるか1であるかの場合のみ同等である。  
【1611】変換  
BitString  
ひとつのアイテムをもつビットストリングは、そのアイテムに等しいビットをつくり出す。  
【1612】Boolean  
偽及び真であるブーリアンはそれぞれ0及び1をつくり出す。  
【1613】Character  
数字0及び1はそれぞれ0及び1をつくり出す。  
【1614】Integer  
整数0及び1はそれぞれ0及び1をつくり出す。  
【1615】Octet  
ビット0を除くすべてのビットが0であるオクテットは、そのビットを作り出す。  
【1616】OctetString  
1ビットに変換可能なオクテットに変換可能なオクテットストリングは、そのビットを作り出す。  
【1617】String  
1ビットに変換可能なキャラクタに変換可能なストリングはそのビットを生成する。  
【1618】4. 7 ビットストリング (Bit String)  
オブジェクト (リファレンスされる)  
・ コレクション  
・ リスト (順序有り)  
・ コンストレインドリスト (コンストレインド)  
・ ビットストリング (実行される)  
クラス  
BitString:  
シールされたインターフェース (ConstrainedList [Bit],

Executed) = (...);

#### 構成

#### initialize

保護されていないop(segments: Object ...

/\*Bit|protected BitString!\*/);

レスポンスを、レスポンス中の位置が左から右にかけて増大しているセグメントの連鎖をつくる（アーギュメント "segment"）。

#### 【1619】アダプテーション

#### constraint:

属性はシールされている。その "ofClass"、"classId"、"isInstance"、"isOptional"、及び "passage" 属性は、クラス "Bit"、クラス "Bit" の識別子、"真"、"偽"、及び "byCopy" である。

#### 【1620】isAfter

短い長さのビットストリングを長いビットストリングに等しい長さとするために十分な数の "0" ビットを短い長さのビットストリングに最初にあらかじめ付加されたかのように1ビットストリングが別のものの後に続く。

#### 【1621】isBefore

短い長さのストリングの長さが長いビットストリングと同じ長さになるのに十分な数の "0" ビットを短いビットストリングに最初に付加されたかのように1ビットのストリングが別なストリングの前にある。

#### 【1622】転換

#### Bit

1ビットはその唯一のアイテムがそのビットに等しいビットストリングを生ずる。

#### 【1623】Boolean

ブーリアンは最初にブーリアンビットに変換したのちそのビットをビットストリングに転換する結果を生ずる。

#### 【1624】Character

キャラクタはオクテットストリングにキャラクタを最初に変換し、次にオクテットストリングをビットストリングに変換することの結果を生ずる。

#### 【1625】Integer

ある整数の "n" について  $[-2^n, 2^n)$  の範囲にあってこれより小さくない整数は、長さ "n" のビットストリングを生ずる。2の補数の表示が用いられる。1の位置であるビットは  $-2$  の  $n$  乗を表す。 $(1, n)$  の範囲内にある各々の位置、"i" にあるビットは、 $2^{n-i}$  を表す。

#### 【1626】Octet

オクテットは、オクテットをオクテットストリングに最初に変換したのち次にオクテットストリングをビットストリングに変換する結果を生じる。

#### 【1627】OctetString

長さが "n" に等しいオクテットストリングは、長さが "8n" に等しいビットストリングを作り出す。ビットストリング中のその位置が  $(8i + j)$  であるビット

は、オクテットストリング中の位置が "i" であるオクテットのビット  $(8 - j)$  である。上記において "i" 及び "j" は整数であり、"i" は、 $[0, n)$  の範囲にあり "j" は  $[1, 8]$  の範囲にある。

#### 【1628】String

ストリングは、ストリングをオクテットストリングに最初に変換し、次にオクテットストリングをビットストリングに変換する結果を生ずる。

#### 【1629】4. 8 ブーリアン (Boolean)

オブジェクト (リファレンスされる)

・ プリミティブ (実行され変更されない)

・ ブーリアン (順序有り)

#### クラス

#### Boolean:

シールされたインターフェース (primitive, Ordered) = (...);

パブリックインスタンスオペレーション

and

op(boolean: Boolean) Boolean;

ブーリアンの論理的な結合 (アーギュメント "boolean") 及びレスポンスをリターンする。

#### 【1630】not

op() Boolean;

レスポンスの論理的な否定をリターンする。

#### 【1631】or:

op(boolean: Boolean) Boolean;

ブーリアンの論理的な択一 (アーギュメント "boolean") 及びレスポンスをリターンする。

#### 【1632】アダプテーション

#### isAfter

第1のブーリアンが "真"、第2のブーリアンが "偽" の場合、前者は後者より後である。

#### 【1633】isBefore

第1のブーリアンが "偽"、第2のブーリアンが "真" の場合、前者は後者より前である。

#### 【1634】isEqual

2つのブーリアンがいずれも "偽" または "真" である場合、両者は等しい。

#### 【1635】変換

#### Bit

ビットは、ビットが1であるか否かを示す。

#### 【1636】Character

キャラクタは、キャラクタのユニコードが0でないか否かを示す。

#### 【1637】Collection

コレクションは、コレクションの長さが0でないか否かを示す。

#### 【1638】Exception

エクセプションは、"真" を生じる。

#### 【1639】Nil

無（ゼロ）は、“偽”を生じる。

【1640】Number

ナンバは、ナンバが0でないか否かを示す。

【1641】Octet

オクテットは、オクテットのビットのいずれかが1であるかを示す。

【1642】4. 9 カレンダ時間 (Calendar Time)

オブジェクト（リファレンスされる）

・ カレンダ時間

クラス

CalendarTime:

Interface = (...);

構成

initialize

レスポンドの本来の属性をなくす。

【1643】パブリックインスタンス属性

day:

Integer|Nil;

レスポンドが特定する日または0。

【1644】dayOfWeek:

リードオンリ Integer|Nil

レスポンドが特定する曜日または0。

【1645】dayOfYear:

リードオンリ Integer|Nil

レスポンドが特定する月日または0。

【1646】dst:

Integer|Nil;

レスポンドが特定するタイムゾーンのパーマネントオフセットからの分における季節的なオフセットまたは0。

【1647】hour:

Integer|Nil;

レスポンドが特定する時または0。

【1648】minute:

Integer|Nil;

レスポンドが特定する分または0。

【1649】month:

Integer|Nil;

レスポンドが特定する月または0。

【1650】second:

Integer|Nil;

レスポンドが特定する秒または0。

【1651】Year:

Integer|Nil;

レスポンドが特定するグレゴリオ歴の年または0。

【1652】zone:

Integer|Nil;

レスポンドが特定するタイムゾーンのUTCからの分におけるパーマネントオフセットまたは0。

【1653】パブリックインスタンスオペレーション

globalize:

unprotected op0;

レスポンドのパーマネント及び季節的なオフセットをUTCオフセットに設定し、特定された絶対時点が変わらないようにする。

【1654】localize:

unprotected op0;

レスポンドのパーマネント及び季節的なオフセットを現在位置のオフセットに設定し、特定された絶対時点が変わらないようにする。

【1655】nomalize:

unprotected op0 Boolean;

レスポンドを規格化し、レスポンドの属性のいずれかがプロセス中変化したか否かをリターンする。

【1656】アダプテーション

isEqual

2つのカレンダー時間は、その本来の属性に従い、同等である。

【1657】変換

Time

タイムは、同一の絶対時点と同一のパーマネント及び季節的なオフセットを特定する規格化されたカレンダー時間を生じる。

【1658】4. 10 ケースド (Cased)

Cased

クラス

Cased:

abstract interface 0 = (...);

パブリックインスタンス属性

isLower:

abstract readonly Boolean;

レスポンドのいずれかのアイテムがローア-ケースか否か。

【1659】isUpper:

abstract readonly Boolean;

レスポンドのいずれかのアイテムがアップア-ケースか否か。

【1660】パブリックインスタンスオペレーション

makeLower:

abstract op0 copied cased;

各々のアップア-ケースのアイテムがそのアイテムのローア-ケースの同等物に置き換えられたレスポンドのコピーをリターンする。

【1661】makeUpper:

abstract op0 copied cased;

各々のローア-ケースのアイテムがそのアイテムのアップア-ケースの同等物に置き換えられたレスポンドのコピーをリターンする。

【1662】4. 11 キャラクタ (Character)

オブジェクト（リファレンスされる）

・ プリミティブ（実行され変更されない）

・ ・ キャラクタ (ケースド、順序付けされる)

クラス

Character:

シールされたインターフェイス (Primitive, Cased, Ordered) = 0;

アダプテーション

isAfter

そのユニコードのコードに従って1つのキャラクタが次のキャラクタの後にある。

【1663】 isBefore

そのユニコードのコードに従って1つのキャラクタが別のキャラクタの前にある。

【1664】 isEqual

2つのキャラクタはそのユニコードのコードに従って同等である。

【1665】 変換

Bit

ビット”0”は数字0を生じビット1は数字1を生ずる。

【1666】 Integer

[0, 65535] の範囲の整数は、そのユニコードのコードのキャラクタを生ずる。

【1667】 Octet

オクテットは、オクテットを整数に最初に変換したのちその整数をキャラクタに変換する結果を生ずる。

【1668】 OctetString

オクテットストリングは、オクテットストリングを最初に整数に変換した後その整数をキャラクタに変換する結果を生ずる。

【1669】 String

1つのアイテムを持つストリングはそのアイテムに等しいキャラクタを生ずる。

【1670】 4. 12 サイテーション (Citation)

オブジェクト (リファレンスされる)

・ サイテーション (順序付けされる)

クラス

Citation

インターフェイス (Object, Ordered) = (...);

構造

initialize

Unprotected op (

title: Identifier!; author: Telename|Nil;

majorEdition, minorEdition: Integer|Nil);

レスポンドの本来の属性を同じような名前のアーギュメント (アーギュメント”author”、”majorEdition”、”minorEdition”及び”title”) にする。

【1671】 パブリックインスタンス属性

author

Telename|Nil;

レスポンドが割り当てられているかまたはそのようなコ

ンストレイントが課せられるならば、レスポンドによって特定されたオーサーのテレネームであり、他の場合は0である。

【1672】 majorEdition

Integer|Nil;

レスポンドが割り当てられているかそれらのコンストレイントが課せられていれば、レスポンドによって特定されたメジャーエディションの番号であり、さもなければ0である。

【1673】 minorEdition

Integer|Nil;

レスポンドが割り当てられているかまたはそのようなコンストレイントが課せられた時、レスポンドによって特定されたマイナーエディションの番号、他の場合には0。

【1674】 title

Identifier!;

レスポンドが特定するタイトルの識別子。

【1675】 アダプテーション

isAfter

その”オーサー (author)” 属性が、仲間であり、それらのものの”タイトル (title)” 属性が、相等しく、第1のサイテーションの”majorEdition” 属性が第2のサイテーションの”majorEdition” 属性の後にあるかまたは両方のサイテーションの”majorEdition” 属性が相等しく第1のサイテーションの”majorEdition” 属性が第2のサイテーションのものの後にある場合にのみ、あるサイテーションが別のサイテーションの後になる。

【1676】 isBefore

2つのサイテーションの”オーサー (author)” 属性が同等であり、”タイトル (title)” 属性が等しく、第1のサイテーションの”majorEdition” 属性が第2のサイテーションの”majorEdition” 属性の前にあるかまたは両方のサイテーションの”majorEdition” 属性が等しく、第1のサイテーションの”minorEdition” 属性が第2のサイテーションの”minorEdition” 属性の前にある場合にのみ、第1のサイテーションは、第2のサイテーションの前になる。

【1677】 isEqual

2つのサイテーションは、それらの”オーサー (author)” 属性が同等で他の固有の属性が等しいとき、等しい。

【1678】 4. 13 サイティド (Cited)

Cited

クラス

Cited:

abstract interface 0 = (...);

構造

initialize:

unprotected op (title: Identifier!;

majorEdition, minorEdition: Integer);  
レスポンドの本来の属性を、その”オーサー”属性が現在のプロセスの割り当てられたテレネームであり、それ以外の属性が同じ様な名前のアーギュメント（アーギュメント”majorEdition”、”minorEdition”及び”title”）であるサイテーションにする。

【1679】パブリックインスタンス属性

citation:

リードオンのプロテクトされたサイテーション;  
レスポンドの割り当てられたサイテーション。

【1680】4. 14 クラス (Class)

オブジェクト (リファレンスされる)

・ クラス (引用され、相互変換される) クラス

Class:

シールされたインターフェース (Object, Cited, Interchanged) = (...);

構成

initialize:

保護されていないop0;

FeatureUnvaliableを投出する。

例外 (“FeatureUnvaliable”) を投出する。

【1681】注: オペレーション”new”でなくオペレーション”makeClasses”を用いてクラスを作り出す。

【1682】パブリックインスタンスオペレーション  
変換

シールされたオペレーション (source: protected Object) copied Objectは、ConversionUnavailableを投出する。

【1683】オブジェクトがすでにそのクラスのインスタンスでない場合、オブジェクト (アーギュメント”source”) をそのクラスに変換することによって生じたレスポンドのインスタンス、または、他の場合にはそのオブジェクトのコピー、をリターンする)。

【1684】オペレーションはソースクラスに最初に変換方法を求め、それがダメな場合に目的点のクラスにその方法を求める。オブジェクトが要求されたように変換できない場合に例外が投出される (“ConversionUnavailable”)。

【1685】注: 2つのステップは、特別の変換のための方法をインプリメントするために、ソースまたは目的点クラスのうちどちらかのよりおそく定義された方を許可またはパーミットする。

【1686】isInstance:

シールされたop(instance: protected Object) Boolean;

オブジェクト (アーギュメント”instance”) がレスポンドのインスタンスであるか否かの表示をリターンする。

【1687】isMember:

シールされたop(member: protected Object) Boolean;

オブジェクト (アーギュメント、”member”) がインターフェースであるが必ずしもインプリメンテーション、レスポンドのメンバ、ではないか否かの表示をリターンする。

【1688】isSubclass:

シールされたop(subclass: Class) Boolean;

クラス (アーギュメント”subclass”) がレスポンドのサブクラスであるか否かの表示をリターンする。

【1689】new:

シールされたop(parameters: Object...) Objectは、ClassAbstract、Exception、objectUninitializedを投出する。

【1690】初期化パラメータ (アーギュメント”parameters”) によって定められたレスポンドの新しいインスタンスをリターンする。前記の署名にもかかわらず、オペレーションの署名は、レスポンドによって定義されたオペレーション”initialize”のそれである。

【1691】レスポンドが抽象的であるか (“ClassAbstract”)、クラス特異の問題が提起されるか (“Exception”) または潜在的なオブジェクトがそれ自身のフィーチャを使用するか (“ObjectUninitialized”) した場合に、例外が投出される。

【1692】4. 15 クラス定義 (Class Definition)

オブジェクト (リファレンスされる)

・ クラス定義

クラス

ClassDefinition:

シールされたインターフェース = (...);

構成

initialize

プロテクトされないop (

title:Identifier!;

majorEdition, minorEdition: Integer;

interface: Interface;

implementation: Implementation|Nil);

レスポンドの本来の属性を、同じような名前のアーギュメント (アーギュメント”implementation”、”interface”、”majorEdition”、”minorEdition”、及び”title”) にする。

【1693】パブリックインスタンス属性

implementation:

Implementation|Nil;

もし、インプリメンテーションがあれば、レスポンドが定義するクラスのインプリメンテーションであり、その他の場合には0である。

【1694】interface:

Interface;

レスポンドが定義するクラスのインターフェース。

【1695】majorEdition:

Integer;

レスポンドが定義するクラスの割り当てられたサイティ  
ションの"majorEdition"属性。

【1696】minorEdition:

Integer;

レスポンドが定義するクラスの割り当てられたサイティ  
ションの"minorEdition"属性。

【1697】title

Identifier!;

レスポンドが定義するクラスの割り当てられたサイティ  
ションの"title"属性。

【1698】パブリックインスタンスオペレーション  
makeClasses:

op(definitions: protected ClassDefinition...)

Lexion [Class]

クラス例外を投出する。

【1699】レスポンドと他のクラス定義（アークグメ  
ント"definitions"）が集散的に定義するクラスのレ  
キシコンをリターンする。各々の値は、そのタイトルが  
関連されたキーに等しいクラスである。サイティション  
は次のように各々のクラスに割り当てられる。割り当て  
られたサイティションの"オーサー（author）"属性  
は、現在のプロセスの割り当てられたテレネームであ  
る。割り当てられたサイティションの他の属性は、クラ  
スのクラス定義によって定義された通りである。

【1700】クラスが作り出されない場合には例外が投  
出される（"classException"）。

【1701】アダプティション

isEqual:

2つのクラスの定義はそれぞれの本来の属性に従って同  
等である。

【1702】4. 16 クラスの例外（Class Exceptio  
n）

オブジェクト（リファレンスされる）

- ・ 例外（変更なし）
- ・ ・ プログラミングの例外
- ・ ・ ・ クラスの例外

クラス

ClassException:

abstract interface (ProgrammingException) = 0;

サブクラス

ClassSealed:

interface (ClassException) = 0;

提案された新しいクラスは、シールされた直接スーパ  
ークラスを有する。

【1703】ClassUndefined:

interface (ClassException) = 0;

提案された新しいクラスは、定義されないクラス識別子  
を使用する。

【1704】FeatureRedefined:

interface (ClassException) = 0;

提案された新しいクラスとそのインターフェーススーパ  
ークラスの1つとは、同一の識別子によって表される本  
来のフィーチャを有する。

【1705】FeatureSealed:

interface (ClassException) = 0;

提案された新しいクラスは、クラスのインターフェース  
スーパークラスの1つによってシールされたフィーチャ  
を実施する。

【1706】FeatureUndefined

interface (ClassException) = 0;

提案された新しいクラスは定義されていないフィーチャ  
識別子を使用する。

【1707】MixinDisallowed:

interface (ClassException) = 0;

提案された新しいクラスは、フレーバが必要とされる場  
合にミックスインを使用する。

【1708】superclassesInvalid:

interface (ClassException) = 0;

提案された新しいクラスは、インCONSISTENTな直接  
スーパークラスを有する。

【1709】4. 17 コレクション（Collection）

オブジェクト（リフェレンスされる）

・ Collection

クラス

Collection:

interface [itemClass: Class] = (...);

サブジェクトクラスの各々のメンバのその各々のアイテ  
ムがそれ自身メンバであるクラス（アークグメント"it  
emClass"）によってパラメータ化される。

【1710】構成

initialize:

プロテクトされないop(item:itemClass...)は、ItemInv  
alidを投出する。

【1711】レスポンドのアイテムをオブジェクトにする  
（アークグメント"items"）。

【1712】提案されたアイテムが不適切である場合に  
例外が投出される（"ItemInvalid"）。

【1713】パブリックインスタンス属性

length:

read-only Integer;

レスポンドの長さ。

【1714】パブリックインスタンスオペレーション

clear:

プロテクトされないop 0;

そのすべてのアイテムをレスポンドから取り除き廃棄す  
る。

【1715】注： オペレーションはレスポンドの長さ  
を0に減少させる。

【1716】examine:

op(item: protected itemClass) itemClass|Nil;

少なくとも1つのそのようなアイテムがあれば、オブジェクト（アーギュメント”item”）に等しいアイテムをレスポンドに残してそれをリターンし、さもなければ0をリターンする。いくつかのそのようなアイテムがあれば選択された1つは定義されないものである。

【1717】exclude:

プロテクトされないop (item: protected itemClass) itemClass|Nil;

少なくとも1つのそのようなアイテムがあれば、オブジェクトに等しいアイテム（アーギュメント”item”）をレスポンドから除去してそれをリターンするかまたさもない場合には0をリターンする。いくつかのそのようなアイテムがある場合には選択された1つは定義されないものである。

【1718】include:

プロテクトされないop (item:itemClass) は、ItemInvalidを投出する。

【1719】オブジェクト（アーギュメント”item”）をレスポンドに新しいアイテムとして含める。

【1720】提案されたアイテムがそのようなものとして不適切であれば例外が投出される（”itemInvalid”）。

【1721】stream:

op () copied Stream[itemClass];

そのアイテムがレスポンドのアイテムであるストリームをリターンする。クラス”Collection”のサブクラスは、ストリームがレスポンドのアイテムを生成させる順序を定義することができれば、クラス”Collection”は、オーダーを規定しないままにしておく。

【1722】ストリームが作り出されたのちにレスポンドが変更された場合には、新しく含められたアイテムは、ストリームが以前にそれらのアイテムを作り出さず、ストリームがもし以前にそれらを作り出さない限り新しく除外されるアイテムを生成させない場合にのみ、ストリームがこれらの新しい含められたアイテムを生成させる。

【1723】注：ストリームがインスタンスであるクラス”ストリーム”のサブクラスは定義されていない。

【1724】アダプテーション

isEqual:

2つのコレクションは、それらのものの長さが等しくそれらのもののアイテムが対をなしており、各々の対のアイテムが同一になるようにされている場合にのみ同等である。コレクションのサブクラスはこの同一性の定義を狭くすることができるが広くすることはできない。

【1725】4. 18 コレクションの例外 (collection Exception)

オブジェクト（リファレンスされない）

・ エクセプション（変更されない）

・ ・ プログラミングの例外

・ ・ ・ Collectionの例外

クラス

CollectionException:

abstract interface (Programming Exception) = 0;

サブクラス

ItemDuplicated:

interface (Collection Exception) = 0;

あるセットの1つの提案されたアイテムは他のものに等しい。

【1726】ItemInvalid:

interface (Collection Exception) = 0;

コレクションの提案されたアイテムは、コレクションのコンストレイントを満たさない。

【1727】KeyDuplicated:

interface (Collection Exception) = 0;

ディクショナリの1つの提案されたキーは、別なものに等しい。

【1728】KeyInvalid:

interface (Collection Exception) = 0;

ディクショナリの提案されたキーは、ディクショナリのコンストレイントに適合しないかまたはディクショナリのキーに等しくなることを意図するオブジェクトはそうにしない。

【1729】ObjectsUnpaired:

interface (Collection Exception) = 0;

ディクショナリの対をなしていると考えられるキー及び値は奇数である。

【1730】PositionInvalid

interface (Collection Exception) = 0;

リストまたはプロシージャにおいてある位置に等しくなることを意図するインテジャは、そのようにしない。

【1731】StackDepleted

interface (Collection Exception) = 0;

スタックの長さはスタックの要求されたマニユプレーションを阻止する。

【1732】4. 19 コンストレインド (Constrained)

Constrained

クラス

Constrained:

abstract interface 0=(...);

構成

initialize

プロテクトされないop (constraint: copied constraint|Nil);

レスポンドの本来の属性を、0でない場合にコンストレイント（アーギュメント”constraint”）にし、その他の場合には初期化パラメータを0のものにする。

【1733】パブリックインスタンス属性



constraint:

リードオンのプロテクトされたコンストレイント;  
ノミネートされたレポンドのプロパティに対するコンス  
トレイント。

【1734】4. 20 コンストレインドディクショナ  
リ (Constrained Dictionary)

オブジェクト (リファレンスされない)

- ・ コレクション
- ・ ・ セット
- ・ ・ ・ ディクショナリ
- ・ ・ ・ ・ コンストレインドディクショナリ (コンス  
トレインド)

クラス

ConstrainedDictionary:

```
interface [KeyClass, valueClass: Class]
(Dictionary [keyClass, valueClass], Constrained) =
0;
```

サブジェクトクラスの各々のメンバの各々のアイテムの  
キー (アーギュメント "keyClass") 及び値 (アーギュ  
メント "valueClass") がそれ自身メンバであるクラス  
によってパラメータ化される。

【1735】構成

initialize:

プロテクトされないop (constraint:copied Constrai  
nt;

keysAndValues: Object ...

/\*key:keyClass;value:valueClass \*/

KeyDuplicated, KeyInvalid, ObjectsUnpairedを送出す  
る。

【1736】キーの値の対 (アーギュメント "keyAndVa  
lues") とレスポンドの "constraint" 属性との間のレス  
ポンドのアイテムの関連を、コンストレイントにする  
(アーギュメント "constraint")。

【1737】2つの提案されたキーが等しい ("keyDup  
licated") か、または提案されたキーがそのようなも  
のとして不適切か ("keyInvalid") またはオブジェク  
トの数が奇数である ("ObjectUnpaired") 場合に例外  
が投出される。

【1738】アダプテーション

add

コンストレイントを満たすキーのみが含まれる。

【1739】constraint

コンストレイントは、キーに対するものである。

【1740】include

コンストレイントを満たすキーのみが含まれる。

【1741】rekey

コンストレイントを満たすキーのみが含まれる。

【1742】set

コンストレイントを満たすキーのみが含まれる。

【1743】4. 21 コンストレインドリスト (Cons

trained List)

オブジェクト (リファレンスされる)

- ・ コレクション
- ・ ・ リスト (順序付けされる)
- ・ ・ ・ Constrained List (Constrained)

クラス

ConstrainedList:

```
interface [itemClass: Class]
(List [itemClass], Constrained) = (...);
```

サブジェクトクラスの各々のメンバの各々のアイテム自  
身がメンバであるクラス (アーギュメント "itemClas  
s") によってパラメータ化される。

【1744】構成

initialize:

プロテクトされないop (constraint:copied Constrai  
nt;

items:itemClass...)

ItemInvalidを投出する。

【1745】レスポンドのアイテムであるオブジェクト  
(アーギュメント "items")、左から右にかけて増大  
するその位置ならびにレスポンドの "constraint" 属性  
をコンストレイントにする (アーギュメント "constrai  
nt")。

【1746】提案されたアイテムが不適切であれば例外  
が投出される ("ItemInvalid")。

【1747】アダプテーション

add

コンストレイントを満たすオブジェクトのみが含まれ  
る。

【1748】constraint

コンストレイントは、アイテムに対するものである。

【1749】include

コンストレイントを満たすオブジェクトのみが含まれ  
る。

【1750】4. 22 コンストレインドセット (Cons  
trained Set)

オブジェクト (リファレンスされる)

- ・ コレクション
- ・ ・ セット (検査される)
- ・ ・ ・ Constrained Set (コンストレインド)

クラス

ConstrainedSet:

```
interface [itemClass:Class]
(Set [itemClass], Constrained) =(...);
```

サブジェクトクラスの各々のメンバの各々のアイテム自  
身がメンバであるクラス (アーギュメント "itemClas  
s") によってパラメータ化される。

【1751】構成

initialize:

プロテクトされないop (constraint: copied Constrai

t;  
item:itemClass...)

ItemDuplicated, ItemInvalidを投出する。

【1752】レスポндаのアイテムであるオブジェクト（アーギュメント”items”）及びレスポндаのコンストレイントである属性をコンストレイントにする（アーギュメント”constraint”）。

【1753】2つの提案されたアイテムが等しい場合（”ItemDuplicated”）または提案されたアイテムがそのようなものとして不適切な場合（”ItemInvalid”）例外が投出される。

【1754】アダプテーション

constraint:

コンストレイントはアイテムに対するものである。

【1755】include:

コンストレイントを満たすオブジェクトのみが含まれる。

【1756】4. 23 コンストレイント (Constraint)

オブジェクト (リファレンスされる)

・ コンストレイント

クラス

Constraint:

interface = (...);

構成

initialize:

プロテクトされないop

(classId, passage: Identifier!|Nil;

isOptional, isInstance: Boolean|Nil)は、PassageInvalidを投出する。

【1757】レスポндаの”ofClass”属性を0にし、レスポндаの他の本来の属性を同じような名前のアーギュメント（アーギュメント”classId”、”isInstance”、”isOptional”、”passage”）とする。アーギュメントが0であれば、対応する属性は、それぞれクラス”Object”、”false”、”false”または”byRef”の識別子とされる。

【1758】提案された通路がそのようなものとして不適切な場合には、例外が投出される（”PassageInvalid”）。

【1759】パブリックインスタンス属性

classId:

Identifier!;

レスポндаのサブジェクトがメンバとなるべきクラスの識別子。識別子は、レスポндаのコンテキストを与えるインターフェースまたはインプリメンテーションの属性”vocabulary”に従って解釈される。

【1760】isInstance:

Boolean;

レスポндаのサブジェクトが属性”classId”が表すク

ラスのインスタンス（単なるメンバではない）となるか否か。

【1761】isOptional:

Boolean;

属性”classId”にかかわらず、レスポндаのサブジェクトが0となることがパーミットされるか否か。

【1762】ofClass:

リードオンリ Class|Nil;

0でない場合、レスポндаのサブジェクトがメンバとなるべきクラス。

【1763】passage:

Identifier!は、PassageInvalidを投出する。;

レスポндаのサブジェクトがどのようにパスされるかを表す識別子。

【1764】提案された通路がそのようなものとして不適切であれば例外が投出される（”PassageInvalid”）。

【1765】アダプテーション

isEqual

2つのコンストレイントはそれぞれの本来の属性に従って相等しい。

【1766】4. 24 コンタクト (Contact)

オブジェクト (参照される)

・ コンタクト

クラス

Contact:

interface = (...);

構成

initialize:

プロテクトされないop (subject:Process|Nil;

subjectNotes:Object|Nil);

レスポндаの”subject”及び”subjectNotes”属性を同じような名前のアーギュメント（アーギュメント”subject”及び”subjectNotes”）及びレスポндаの他の本来の属性を、”subject”属性がプロセスであればサブジェクトを記述するものに、またその他の場合には0にする。

【1767】パブリックインスタンス属性

Subject:

Process|Nil;

レスポндаのサブジェクトまたは0。

【1768】subjectClass:

リードオンリのプロテクトされたサイテーション|0;

レスポндаのサブジェクトは1つのインスタンスであるクラス”Agent”またはクラス”Place”のサブクラスであるクラスの割り当てられたサイテーションは、”subject”属性がプロセスであるかまたはそのクラスであり、”subject”属性が0であれば0である。

【1769】subjectName:

リードオンリのプロテクトされたtename|Nil;

” subject” 属性がプロセスであれば、レスポンドのサブジェクトの割り当てられたテレネームに〔またはそのテレネーム〕またはその他の場合には0である。

【1770】 subjectNotes:

Object|Nil;

レスポンドのオブザーバによって保持されたオブジェクト。

【1771】 アダプティション

isEqual:

2つのコンタクトは、それらのものの属性” subjectName” に従って同等である。

【1772】 4. 25 コンタクティド (Contacted)

Contacted

クラス

Contacted:

abstract interface () = (...);

構成

initialize:

レスポンドの属性” contacts” をクリアする。

【1773】 プライベートインスタンスの属性

contacts:

リードオンリ Set [Contact];

レスポンドが保持するコンタクトのセット。

【1774】 4. 26 ディクショナリ (Dictionary)

オブジェクト (リファレンスされる)

- ・ コレクション
- ・ ・ セット
- ・ ・ ・ ディクショナリ

クラス

Dictionary:

interface [keyClass, valueClass: Class]

(set [Association [keyClass, valueClass]]) = (...);

サブジェクトクラスの各々のメンバの各々のアイテムのキー (アーギュメント” keyClass”) 及び値 (アーギュメント” ValueClass”) 自身がメンバであるクラスによってパラメータ化される。

【1775】 構成

initialize:

プロテクトされないop (keysAndValues: Object...

/\*key:keyClass;value:valueClass\*/

keyDuplicated, keyInvalid, ObjectsUnpairedを投出する。

【1776】 キーの値の対の間のレスポンドのアイテムの関連性を作る (アーギュメント” keyAndValues”)。

【1777】 2つの提案されたキーが等しい場合 (“keyDuplicated”)、提案されたキーがそのようなものとして不適切な場合 (“keyInvalid”) またはオブジェクトの数が奇数の場合 (“ObjectsUnpaired”) 例外が投出される。

【1778】 パブリックインスタンスオペレーション  
add:

プロテクトされないop (key:keyClass; value: valueClass)

は、keyInvalidを投出する。

【1779】 レスポンドには、新しい関連、第1のオブジェクトに関係したもの (アーギュメント” key”) が関連のキーとしてそして第2のオブジェクト (アーギュメント” value”) が関連の値としてそれぞれ含まれる。

【1780】 提案されたキーがそのようなものとして不適切な場合またはレスポンドがすでに含めた関連の中にあるキーに等しい場合 (“keyInvalid”) 例外が投出される。

【1781】 drop:

プロテクトされないop (key: protected keyClass) valueClassは、keyInvalidを投出する。

【1782】 キー (アーギュメント” key”) がオブジェクトに等しいアソシエーション (関連) をレスポンドから除外して廃棄して、関連の値をリターンする。主張されたキーがそのようなものとして不適切な場合には例外が投出される (“KeyInvalid”)。

【1783】 find:

op (value: protected valueClass) keyClass|Nil;

その値がオブジェクトに等しい関連 (アーギュメント” value”) をレスポンドに残し、関連のキーをリターンする。レスポンドがいくつかのこのような関連を含む場合、選択されたものは定義されない。レスポンドが0を含む場合、0がリターンされる。

【1784】 get:

op(key: protected keyClass) valueClass

keyInvalidを送出する;

そのキーがオブジェクト (アーギュメント” key”) に等しい関連をレスポンドに残し、関連の値をリターンする。

【1785】 主張されたキーがそのようなものとして不適切な場合には、 (“KeyInvalid”) 例外が投出される。

【1786】 rekey:

プロテクトされないop (currentKey:protected keyClass;

newKey: keyClass)

は、keyInvalidを投出する。

【1787】 そのキーが第1のオブジェクト (アーギュメント” currentKey”) に等しい関連をレスポンドに残し、その関連のキーを廃棄し、そのキーの代わりに、第2のオブジェクト (アーギュメント” newKey”) を代替する。このオペレーションはオペレーション” drop” 及び” add” によるかのように行う。

【1788】 主張されるかまたは提案されたキーがその

ようなものとして不適切であるかまたは、この後者のキーがレスポндаにすでに含まれている関連中のキーに等しい場合に、例外が投出される (" KeyInvalid" )。

【1789】set:

プロテクトされないop (key: protected keyClass; value:

valueClass)

は、KeyInvalidを投出する。

【1790】そのキー (アークグメント "key") が第1のオブジェクトに等しいアソシエーションをレスポндаが含む場合 (アークグメント "key")、そのアソシエーションの代わりに、第2のオブジェクトを代替する (アークグメント "value")。その他の場合には、レスポндаに新しいアソシエーションを含める。第1のものは、第1オブジェクト (アークグメント "key") をアソシエーションのキーとして含むものであり、第2のもの (アークグメント "value") はそれをアソシエーションの値として含むものである。

【1791】提案されたキーがそのようなものとして不適切な場合には、例外が投出される (" KeyInvalid" )。

【1792】transpose:

プロテクトされないop (key 1, key 2: protected keyClass)

は、KeyInvalidを送出する;

2つのキー (アークグメント "key 1" 及び "key 2") の代わりにレスポндаの値を転置する。

【1793】主張されたキーがそのようなものとして不適切な場合には例外が投出される (" KeyInvalid" )。

【1794】アダプテーション

include

ただ1つのアソシエーションのみが含まれる。あるアソシエーションは、アソシエーションに等しい現存のアイテムを排除し、廃棄したのちに新しいアイテムとして含まれる。

【1795】isEqual

その同じようなキーをもつ値が相等しい場合にのみ2つのディクショナリは同等である。

【1796】4. 27 例外 (Exception)

オブジェクト (参照される)

・ 例外 (変更なし)

クラス

Exception:

abstract interface (Object, Unchanged) = (...);

パブリックインスタンスオペレーション

sealed op 0 を投出する。

【1797】レスポндаのためにその方法を失敗に終わらせることによって現在の方法の遂行を終了する。

【1798】アダプテーション

isEqual

2つの例外は、同一のクラスのインスタンスである場合にのみ相等しい。

【1799】4. 28 実行される (Executed)

Executed

クラス

Executed:

abstract interface 0 = 0;

このクラスは、シールされている。

【1800】パブリックインスタンスオペレーション catch:

シールされたop (expection: Class) Exception|Nil

は、例外を投出する。

【1801】レスポндаを遂行し、レスポндаが成功すれば、0をリターンし、例外が供給されたクラスの1つのメンバであれば (アークグメント "exception") 例外をリターンし、レスポндаを失敗させる。レスポндаは、レスポндаのフレーム及びプロパティではなくリクエストのフレーム及びプロパティにアクセスする。

【1802】レスポндаが供給されたクラスのメンバではない例外を投出した場合に、例外が投出される (" Exception" )。

【1803】注: クラスは、例外であるかまたはそのサブクラスである。

【1804】do:

シールされたop 0

は、例外を投出する。

【1805】レスポндаを遂行する。レスポндаはレスポндаのフレーム及びプロパティではなくリクエストのフレーム及びプロパティにアクセスする。

【1806】レスポндаがそのようにした場合、例外が投出される (" Exception" )。

【1807】either:

シールされたop (false: Executed; precondition: Boolean)

は、例外を送出する。:

プレコンディション (アークグメント "precondition") が真であれば、レスポндаを遂行し、そうでなければ遂行されるオブジェクト (アークグメント "false") を遂行する。選択され実行されたオブジェクトは、レスポндаのフレーム及びプロパティではなくリクエストのフレーム及びプロパティにアクセスする。

【1808】選択され実行されたオブジェクトがそのようにする場合、例外が投出される (" Exception" )。

【1809】if:

シールされたop (precondition: Boolean)

は、例外を送出する。;

プレコンディションが "真" である場合にのみレスポндаを遂行する (アークグメント "precondition")。レスポндаはそれ自身のものではなくリクエストのフレーム及びプロパティにアクセスする。

【1810】レスポンドがそのようにする場合、例外が投出される ("Exception")。

【1811】loop:

シールされたop 0

は、例外を投出する。

【1812】レスポンドを不特定回数遂行し、レスポンドが失敗すれば、進めることなく、例外を投出する。レスポンドはレスポンドのフレーム及びプロパティではなく、リクエストのフレーム及びプロパティにアクセスする。

【1813】レスポンドがそのようにする場合、例外が投出される ("Exception")。

【1814】repeat:

シールされたop (repetitions: Integer)

は、例外を送出する。;

要求された回数だけレスポンドを遂行し (アーギュメント "repetitions")、レスポンドが失敗すれば進めることなく例外を投出する。整数が負数であればオペレーションは効果を持たない。レスポンドは、レスポンドのフレーム及びプロパティではなく、リクエストのフレーム及びプロパティにアクセスする。

【1815】オペレーションは、レスポンドの各々の遂行の前に、以前の遂行の数プラス1である整数をスタックに保存する。

【1816】レスポンドがそのようにする場合例外が投出される ("Exception")。

【1817】注: オペレーションはそれがスタックに保存するいかなる整数もポップしない。

【1818】while:

シールされたop (precondition: Executed)

は、Exception, ResultInvalid, ResultMissingを送出する。

【1819】実行されるオブジェクトを繰り返し遂行し (アーギュメント "precondition")、スタックからブーリアンをポップし、ブーリアンが真であれば、レスポンドを遂行する。ブーリアンが偽であればオペレーションは成功する。実行されたオブジェクトが失敗すれば、オペレーションは進めることなく例外を投出する。実行されるオブジェクトはレスポンドのフレーム及びプロパティではなく、リクエストのフレーム及びプロパティにアクセスする。

【1820】実行されたオブジェクトがそのようにする場合 ("Exception")、ポップされるオブジェクトがブーリアンでない場合 ("ResultInvalid") またはポップされるべきオブジェクトがスタックに無い場合 ("ResultMissing") に例外が投出される。

【1821】4. 29 実行の例外 (Execution Exception)

オブジェクト (参照される)

・ 例外 (変更なし)

・ ・ プログラミングの例外

・ ・ ・ カーネル例外

・ ・ ・ ・ Execution Exception

クラス

ExecutionException:

abstract interface (KernelException) = 0;

サブクラス

ArgumentInvalid:

interface (ExecutionException) = 0;

アーギュメントはアーギュメントのコンストレイントを満たさない。

【1822】ArgumentMissing:

interface (ExecutionException) = 0;

アーギュメントは、スタックにない。

【1823】AttributeReadOnly:

interface (ExecutionException) = 0;

属性は、リードオンリであるのセットできない。

【1824】ClassUnavailable:

interface (ExecutionException) = 0;

クラスは、現在のプロセスの属性 "privateClasses" でも、現在のプロセスの属性 "publicClasses" でもない。

【1825】EscalationInvalid:

interface (ExecutionException) = 0;

フィーチャは、不適切にエスカレートされる。

【1826】FeatureUnavailable:

interface (ExecutionException) = 0;

識別子は、レスポンドのアクセス可能なフィーチャを表さない。

【1827】InternalException:

interface (ExecutionException) = 0;

エンジンは、命令セットを十分にインプリメントし得ない。

【1828】PropertyUnderfined:

interface (ExecutionException) = 0;

識別子は、現在の方法に適切に検出可能を意味しない。

【1829】ReferenceProtected:

interface (ExecutionException) = 0;

あるオブジェクトへのリファレンスはそのオブジェクトの変更を防止しない。

【1830】ReferenceVoid:

interface (ExecutionException) = 0;

リファレンスは、ボイドである。

【1831】ResponderMissing:

interface (ExecutionException) = 0;

スタックがクリアされた時にあるフィーチャは、リクエストされる。

【1832】ResultInvalid:

interface (ExecutionException) = 0;

ある結果は、その結果のコンストレイントを満たさな

い。

【1833】ResultMissing:

interface (Execution Exception)=0;

ある結果は、スタックに無い。

【1834】VariableUnderfined:

interface (Execution Exception)=0;

識別子は、変数を表さない。

【1835】4. 30 フィーチャ (Feature)

オブジェクト (参照される)

- ・ フィーチャ

クラス

Feature:

abstract interface = (...);

このクラスは、シールされる。

【1836】構成

initialize:

プロテクトされないop (isPublic: Boolean|Nil;

exceptions: Set [Identifier] |Nil);

レスポンドの本来の属性を同じ名前のアーギュメント (アーギュメント "exceptions" 及び "isPublic") とする。しかしアーギュメントが0であれば、対応する属性はクリアされるかまたは "偽" とされる。

【1837】パブリックインスタンス属性

exceptions:

Set [Identifier!];

レスポンドが定義するフィーチャによって投出された例外がそのメンバであるクラスを表す識別子。識別子は、レスポンドを含むインターフェースの属性 "vocabulary" に従って解釈される。

【1838】isPublic:

ブーリアン;

レスポンドが定義するフィーチャがパブリックであるか否か、すなわちプライベートでないか否か。

【1839】4. 31 ハッシュド (Hashed)

Hashed

クラス

Hashed:

abstract interface () = (...);

パブリックインスタンス属性

hash:

abstract read-only Integer;

レスポンドのハッシュ

4. 32 アイデンティファイヤ (Identifier)

オブジェクト (参照される)

- ・ プリミティブ (実行され、変更されない)
- ・ 識別子 (順序有り)

クラス

Identifier:

シールされたinterface (Primitive, Ordered) = 0;

アダプテーション

isAfter

1つの識別子は、そのテキストに従って別の識別子の後にある。

【1840】isBefore

1つの識別子は、そのテキストに従って別のものの前にある。

【1841】isEqual

2つの識別子は、そのテキストに従って同一である。

【1842】変更

String

あるストリングは、識別子のテキストがその識別子を形成するので適切である。

【1843】4. 33 インプレメンテーション (Implementation)

オブジェクト (参照される)

- ・ Implementation

クラス

Implementation:

シールされたinterface = (...);

構成

initialize:

プロテクトされないop (superclasses: List [Identifier!] |Nil;

vocabulary: Lexion [Citation] |Nil;

property: List [Identifier!] |Nil;

instanceMethods, setMethods, classMethods,

fromMethods, toMethods: Lexicon [Method] |Nil;

レスポンドの本来の属性を同じ名前のアーギュメント

(アーギュメント "classMethods", "fromMethods", "instanceMethods", "properties", "setMethods", "superClasses", "toMethods" 及び "vocabulary") とする。アーギュメントが0であるが、対応する属性は0であることが許されない場合に、対応する属性はクリアされる。

【1844】パブリックインスタンス属性

classMethods:

Lexicon [Method];

メソッド-許可されない "αSet" オペレーション

以外のオペレーション "αGet" を含むクラスオペレーション-レスポンドを組み込むクラスに対して固有のもの。各々の値は、関連するキーが表すオペレーションを遂行するための方法である。

【1845】fromMethods:

Lexicon [Method];

レスポンドを組み込んだクラスに対して固有の方法-他のクラスから変更するための方法。各々の値は、関連するキーが表すクラスから変更するための方法である。

【1846】注: クラス "Dictionary" の実施 (インプレメンテーション) において、特定されるクラス "List" であれば、変換は、そのレスポンドがクラス "Dicti

onary”でありその署名が”List”オペレーション”  
(List : protected list) Dictionary”である変換で  
ある。

instanceMethods:

Lexicon [Method];

オペレーション”αセット”(属性”setMethods”)以外  
のオペレーション”αゲット”を含むインスタンスオ  
ペレーションのためのレスポンドを組み込んだクラス  
に固有の方法。各々の値は、関連するキーを表すオペ  
レーションを遂行するための方法である。

【1847】properties:

List [Identifier!];

レスポンドを組み込むクラスに固有のプロパティの識別  
子。これらの識別子の順序は、開示された命令セットの  
解釈において意味を持たない。しかしプロパティの順序  
は、この開示のアペンディクスBに示された符号化規則  
において大切である。

【1848】setMethods:

[Lexicon [Method];

レスポンドを組み込んだクラスに固有の”αSet”イ  
ンスタンスオペレーションのための方法。各々の値は  
関連するキーを表すオペレーションを遂行する方法であ  
る。

【1849】superclasses:

List [Identifier!]: Nil;

クラスのインターフェイスの直接スーパークラスと異な  
っているならば、レスポンドを組み込んだクラスのイン  
プリメンテーション直接スーパークラスの識別子であ  
り、さもなければ0である。クラスの標準的な順序は、  
その識別子の順序であり、最後のクラスはフレーバとな  
るべきである。

【1850】toMethods:

Lexicon [Method];

レスポンドを組み込んだクラスに固有の他のクラスに  
変換するための方法。各々の値は関連するキーが表す  
クラスに変換するための方法である。

【1851】注:クラス”Dictionary”のインプリメン  
テーションにおいて、特定されるクラスがクラス”Lis  
t”であれば、変換は、そのレスポンドがディクショナ  
リであり、その署名が”List:op () List”である変  
換である。

【1852】vocabulary:

Lexicon [Citation]: Nil;

クラスのインターフェイスが参照するものでなければ、  
レスポンドが参照するユーザによって特定されたクラス  
であり、さもなければ0である。各々の値は関連するキ  
ーを表すクラスへのサイテーションである。属性は、レ  
スポンドの”fromMethods”、”superClass”、及び”t  
oMethods”属性(他のものではない)の解釈を案内す  
る。

【1853】アダプテーション

isEqual

2つのインプリメンテーションはその本来の属性に従っ  
て同一である。

【1854】4. 3 4 整数(Integer)

オブジェクト(参照される)

・ プリミティブ(実行され変更されない)

・ 数(順序あり)

・ Integer

クラス

Integer:

シールされたインターフェイス (Number)=(...);

パブリックインスタンスオペレーション

modulus:

op (divisor: Integer) Integer

は、DivisionByZeroを投出する;

レスポンドの算数の剰余、被除数と整数(アーギュメン  
ト”divisor”)即ち除数とをリターンする。前者の符  
号は結果を表す。

【1855】除数が0である場合に例外が投出され  
る(”DivisionByZero”)。

【1856】quotient:

op (divisor: Integer) Integer

は、DivisionByZeroを投出する;

レスポンドの算数の商、被除数と整数(アーギュメン  
ト”divisor”)、除数とリターンする。

【1857】除数が0である場合、例外が投出され  
る(”divisionByZero”)。

【1858】アダプテーション

abs

結果は整数である。

【1859】add

アーギュメントが実数であれば結果は実数であり、整数  
であれば結果は整数である。

【1860】divide

アーギュメントが実数であれば結果は実数であり、アー  
ギュメントが整数であり結果が正しいと思われれば結果  
は整数であり、さもなければ結果は実数である。

【1861】multiply

オペレーション”add”について前述した通り。

【1862】negate

結果は整数である。

【1863】subtract

オペレーション”add”について前述した通り。

【1864】変換

Bit

ビット”0”及び”1”は其々0及び1を生ずる。

【1865】BitString

ある整数の変換によって生成され得るビットストリング  
は整数を生ずる。

## 【1866】 Boolean

ブーリアン”偽”及び”真”は其々0及び1を生ずる。

## 【1867】 Character

あるキャラクタは、そのキャラクタのユニコードコードである[0, 65535]中の整数を生ずる。

## 【1868】 Octet

オクテットはオクテットをオクテットストリングに最初に変更し次にオクテットストリングを整数に変更することの結果を生ずる。

## 【1869】 OctetString

ある整数を変換することによって生成させ得るオクテットストリングは整数を生ずる。

## 【1870】 Real

実数は、オペレーション”truncate”を遂行することによって結果した整数を生ずる。

## 【1871】 String

キャラクタテレスクリプトにおいてトークン”Integer”のためのシンタクティックルールに従うストリングは、対応する整数を生ずる。

【1872】 4. 35 インターチェンジド (Interchanged)

変更しない

・ Interchanged

クラス

Interchanged:

abstract interface (Unchanged) = (...);

パブリックインスタンス属性

digest:

abstract readonly protected object : Nil;

レスポンド即ちクラスのこの特別のインスタンスが相互変換可能であれば、レスポンドのダイジェストであり、さもなければ0である。

【1873】 4. 36 インターフェース (Interface)

オブジェクト (参照される)

・ Interface

クラス

Interface:

シールされたinterface = (...);

構成

initialize:

プロテクトされないop (superclasses: List [Identifier!]: Nil;

vocabulary: Lexicon [Citation]: Nil;

instanceFeatures: Lexicon [Feature]: Nil;

sealedInstanceFeatures: Set [Identifier!]: Nil;

classFeatures: Lexicon [Feature]: Nil;

sealedClassFeatures: Set [Identifier!]: Nil;

isAbstract: Boolean: Nil);

レスポンドの固有の属性を同じ名前のアーギュメント

(アーギュメント”classFeatures”、”instanceFeatures”、”isAbstract”、”sealedClassFeatures”、”sealedInstanceFeatures”、”superClass”及び”vocabulary”)とする。しかし対応するアーギュメントが0であれば、属性”superclasses”はクラス”Object”のみの識別子のリストとされ、または、属性が属性”isAbstract”、”superClasses”以外であれば、属性はクリアされる。

【1874】 パブリックインスタンス属性

classFeatures:

Lexicon [Features];

レスポンドを組み込むクラスに固有のクラスのフィーチャ。各々の値は、関連するキーが表すフィーチャ定義である。

【1875】 instanceFeatures:

Lexicon [Feature];

レスポンドを組み込むクラスに固有のインスタンスフィーチャ。各々の値は、関連するキーが表すフィーチャ定義である。

【1876】 isAbstract:

Boolean;

レスポンドを組み込んだクラスがアブストラクトであるか否か。

【1877】 sealedClassFeatures:

Set [Identifier!];

クラスフィーチャの識別子。これらの識別子は、レスポンドを組み込んだクラスに固有かまたはそのクラスによって受け継がれており、そのクラスによってシールされている。

【1878】 sealedInstanceFeatures:

Set [Identifier!];

インスタンスフィーチャの識別子。これらの識別子は、レスポンドを組み込んだクラスに固有であるかまたはこのクラスによって受け継がれて、このクラスによってシールされている。

【1879】 superclasses:

List [Identifier!];

レスポンドを組み込んだクラスのインターフェースイミディエートスーパークラスの識別子。クラスの標準的な順序は、その識別子の順序であり、最後のクラスはフレーバでなければならない。

【1880】 vocabulary:

Lexicon [Citation];

レスポンドが参照するユーザによって定義されたクラス。

【1881】 各々の値は関連するキーが表すクラスへのサイテーションである。属性は、レスポンドの属性”スーパークラス” (他のものではない) の解釈を案内する。

【1882】 アダプテーション



isEqual

二つのインターフェースは、その固有の属性に従って同一である。

【1883】4 37. カーネルの例外 (Kernel Exception)

オブジェクト (参照される)

- ・ 例外 (変更されない)
- ・ ・ プログラミングの例外
- ・ ・ ・ Kernel Exception

クラス

Kernel Exception:

abstract interface (Programming Exception) = 0;

サブクラス

ClassAbstract:

interface (KerneK Exception) = 0;

オブジェクトの提案されたクラスは抽象的である。

【1884】ConversionUnavailable:

interface (Kernel Exception) = 0;

オブジェクトの提案されない変換は定義されない。

【1885】CopyUnavailable:

interface (Kernel Exception) = 0;

オブジェクトのプロパティはコピーできない。

【1886】LoopMissing:

interface (Kernel Exception) = 0;

あるループは破断されまたはループがない場合には連続している。

【1887】MarkMissing:

interface (Kernel Exception) = 0;

スタックのアイテムはマークを含まない。

【1888】ObjectUninitialized:

interface (Kernel Exception) = 0;

オペレーション "initialize:ド" をエスカレートする前にあるオブジェクトはそのフィーチャの1つを使用することを試みる。

【1889】PassageInvalid:

interface (カーネル例外) = 0;

あるオブジェクトに提案された通路は不適切である。

【1890】SelectorDuplicated:

interface (Kernel Exception) = 0;

2つのセレクトは同一である。

【1891】4. 38 レキシコン (Lexicon)

オブジェクト (参照される)

- ・ コレクション
- ・ ・ セット
- ・ ・ ・ ディクショナリ
- ・ ・ ・ ・ コンストレインド ディクショナリ (コンストレインド)
- ・ ・ ・ ・ ・ Lexicon

クラス

Lexicon:

interface (ValueClass: Class]

(constrainedDictionary [Identifier, valueClass]) = (...);

サブジェクトクラスの各々のメンバの各々のアイテムの値それ自身がメンバであるクラス (アーギュメント "valueClass") によってパラメータ化される。

【1892】構成

initialize:

プロテクトされないop (keysAndValues: Object...

/\* key: Identifier!; value: valueClass \*/)

KeyDuplicated, KeyInvalid, ObjectsUnpairedを送出する。;

レスポンドのアイテムを、キーの値対の間のアソシエーションとする (アーギュメント "keyAndValues")。

【1893】2つの提案されたキーが等しく、("keyDuplicated")、提案されたキーがそれ自体として不適切である ("keyInvalid") か、またはオブジェクトの数が奇数である ("ObjectsUnpaired") 場合に例外が投出される。

【1894】アダプテーション

constraint

属性は、シールされている。属性 "ofClass"、"classId"、"isInstance"、"isOptional"、及び "passage" 属性はそれぞれ、"Identifier" クラス、"クラス" "Identifier"、"真"、"偽" 及び "byCopy" の識別子である。

【1895】変換

Dictionary

そのキーがレキシコンのコンストレイントを満たすディクショナリはレキシコンを生ずる。

【1896】4. 39 リスト (List)

オブジェクト (参照される)

- ・ コレクション
- ・ ・ リスト (順序あり)

クラス

List:

interface [itemClass: Class]

(Collection [itemClass], Ordered) = (...);

サブジェクトクラスの各々のメンバの各々のアイテムそれ自身がメンバであるクラス (アーギュメント "itemClass") によってパラメータ化される。

【1897】構成

initialize:

プロテクトされないop (items: itemClass...);

レスポンドのアイテムを、その位置が左から右にかけて増大するオブジェクト (アーギュメント "items") とする。

【1898】パブリックインスタンスオペレーション

add:

プロテクトされないop (position: Integer; item: ite

mClass)

は、ItemInvalid, PositionInvalidを送出する；

新しいアイテムとしてレスポнда中にオブジェクト（アーギュメント”item”）を含める。このアイテムは、その位置が整数（アーギュメント”position”）に等しいアイテムである。含められたアイテムの位置に等しいかまたはそれに後続する位置を持つ各々のアイテムの位置は1つずつ増大する。

【1899】提案されたアイテムがそのようなものとして不適切である場合（”ItemInvalid”）、または提案された位置がそのようなものとして不適切である場合（”PositionInvalid”）例外が送出される。

【1900】drop:

プロテクトされないop (position: Integer) itemClass

は、PositionInvalidを送出する。；

その位置が整数に等しいアイテムをレスポндаから除いてリターンする（アーギュメント”position”）。除去されたアイテムの位置に続く位置の各々のアイテムの位置は1つずつ減少する。

【1901】主張された位置がそのようなものとして不適切な場合に例外が投出される（”PositionInvalid”）。

【1902】find:

op (initialPosition: Integer; item: protected itemClass)

Integer: Nil

は、PositionInvalidを投出する；

オブジェクトに等しく（”item”）その位置が整数の前にないアイテムの位置をレスポндаに残しリターンする（”initialPosition”）。レスポндаが幾つかのそのようなアイテムを持つ場合、その位置が最小のものを選ぶ。何も持たなければ0をリターンする。主張された位置がそのようなものとして不適切な場合（”PositionInvalid”）例外が投出される。

【1903】get:

op (position: Integer) itemClass

は、PositionInvalidを送出する；

その位置が整数に等しいアイテムをレスポндаに残しリターンする（アーギュメント”position”）。主張された位置がそのようなものとして不適切な場合（”PositionInvalid”）例外が投出される。

【1904】reposition:

プロテクトされないop (currentPosition, newPosition: Integer)

は、PositionInvalidを送出する；

その位置が第1の整数に等しいレスポндаのアイテムの位置（アーギュメント”currentPosition”）を第2の整数（アーギュメント”newPosition”）に変更する。このオペレーションは、第2の整数がオペレーション”

drop”のリクエストされる後ではなく前に解釈されることを除いて、オペレーション”drop”及び”add”によるかのようにこれを達成する。

【1905】主張された位置がそのようなものとして不適切な場合（”PositionInvalid”）例外が投出される。

【1906】set:

プロテクトされないop (position: Integer; item: itemClass)

は、ItemInvalid, PositionInvalidを送出する；

アイテムがもしあれば、即ちその位置が整数に等しい（アーギュメント”position”）新しいアイテムが附加されていない場合、レスポндаからそのアイテムを除去し、新しいアイテムとしてオブジェクト（アーギュメント”item”）を同じ位置に含める。

【1907】提案されたアイテムがそのようなものとして不適切な場合（”ItemInvalid”）または提案された位置がそのようなものとして不適切な場合（”PositionInvalid”）例外が投出される。

【1908】transpose:

プロテクトされていないop (position1, position2: Integer)

PositionInvalidを送出する。；

2つの位置を占めるレスポндаのアイテム（アーギュメント”position1”及び”position2”）を転置する。主張された位置がそのようなものとして不適切な場合（”PositionInvalid”）例外が投出される。

【1909】アダプテーション

exclude

オペレーションは、その位置が除外されたアイテムの位置の後になる各々のオブジェクトの位置を1つだけ減少させる。

【1910】include

新しく含められたアイテムの位置はリストの新しい長さである。

【1911】isEqual

2つのリストはその同じ位置にあるアイテムが同一である場合にのみ同一である

stream

ストリームは位置の増大の順序にリストのアイテムを生成させる。

【1912】変換

Procedure

プロシーダは、そのアイテムの各々がプロシーダ中の同じ位置のアイテムのコピーである同じ長さのリストを生ずる。

【1913】4. 40 マーク (Mark)

オブジェクト (参照される)

- ・ プリミティブ (実行され変更されない)
- ・ ・ Mark

クラス

Mark:

シールされたインターフェース (Primitive)=0;

アダプテーション

isEqual

いかなる2つのマークも同等である。

【1914】4. 41 手段 (Means)

オブジェクト (参照される)

- ・ Means

クラス

Means:

abstract interface=0;

4. 42 ミーティングの例外 (Meeting Exception)

オブジェクト (参照される)

- ・ 例外 (変更なし)
- ・ ・ Meeting Exception

クラス

MeetingException:

abstract interface (Exception)=0;

サブクラス

MeetingDenied:

interface (Meeting Exception)=0;

ペティションにはペティションにミーティングを拒絶する。

【1915】MeetingDuplicated:

interface (Meeting Exception)=0;

ペティションとペティションとは同一であるかまたはすでにミーティングである。

【1916】MeetingInvalid:

interface (Meeting Exception)=0;

コンタクトはミーティングを表わさない。

【1917】PetitionExpired:

interface (Meeting Exception)=0;

ペティションには許可された時間内においてミーティングできない。

【1918】4. 43 ミーティングプレイス (Meeting Place)

オブジェクト (参照される)

- ・ プロセス (名前付けされている)
- ・ ・ プレイス (移動しない)
- ・ ・ ・ Meeting Place

クラス

MeetingPlace:

abstract interface (Place) = (...);

パブリックインスタンスオペレーション

meet:

プロテクトされないop (petition: copied Petition) Contact

MeetingException, ProcessNotCurrent, StateImproper  
を送出する。; オペレーションをリクエストしたエー

ジェントとペティション (アーギュメント "petition") が特定するエージェントとの間のミーティングを開始させ、その "subject" 属性が後者であるコンタクトをリターンする。後者のエージェントは常にレスポンドを占有していなければならない。

【1919】ミーティングが失敗に終わり ("MeetingException")、レスポンドが現在のプレイスでなく ("ProcessNotCurrent") またはリクエストのステイトが "meet" を排除する ("StateImproper") 場合に例外が投出される。

【1920】part:

プロテクトされない op (contact: Contact)

は、MeetingInvalid, ProcessNotCurrent, StateImproperを送出する。;

オペレーションをリクエストするエージェントとコンタクトのサブジェクト (アーギュメント "contact") との間のミーティングを終了させ、コンタクトの "subject" 属性を0とする (属性がすでに0となっていない場合)。サブジェクトはやはりレスポンドを占有していなければならない。コンタクトがミーティングのコンタクトでなく ("MeetingInvalid")、レスポンドが現在のプレイスでなく ("ProcessNotCurrent") またはリクエストのステイトがオペレーション "part" を除外する ("StateImproper") 場合、例外が投出される。

【1921】partAll:

プロテクトされないop 0

は、ProcessNotCurrent, StateImproperを送出する; リクエストするエージェントを含む全てのミーティングを終了させた後エージェントをアイソレートする。レスポンドが現在のプレイスでなく ("ProcessNotCurrent") またはリクエストのステイトがオペレーション "partAll" ("StateImproper") を除外する場合、例外が送付される。

【1922】4. 44 メソッド (Method)

オブジェクト (参照される)

- ・ Method

クラス

Method:

シールされたinterface=(...);

構成

initialize:

プロテクトされないop (procedure: Procedure:Nil;

variables: List [Identifier!]:Nil);

レスポンドの本来の属性を同じ名前のアーギュメント (アーギュメント "procedure" 及び "variables") とする。しかしアーギュメントが0であれば対応する属性はクリアする。

【1923】パブリックインスタンス属性

procedure:

Procedure;

レスポンドのプロシージャ。

【1924】variables:

List [Identifier!];

レスポンドの変数の識別子。これらの識別子の順序は開示された命令セットの解釈に関連して重要ではない。しかしこの開示のアペンディクスBに示された符号化規則においては変数の順序は大切である。

【1925】アダプテーション

isEqual

2つの方法はその固有の属性に従って同一である。

【1926】4. 45 その他の例外 (Miscellaneous Exception)

オブジェクト (参照される)

- ・ 例外 (変更されない)
- ・ ・ プログラミング例外
- ・ ・ ・ Miscellaneous Exception

クラス

MiscellaneousException:

abstract interface (ProgrammingException) = 0;

サブクラス

PatternInvalid:

interface (MiscellaneousException) = 0;

パターンの提案されたテキストはシンタクティックに誤っている。

【1927】SeedInvalid:

interface (MiscellaneousException) = 0;

ランダムストリームの提案されたシードが要求された範囲にない。

4. 46 モディファイヤ (Modifier)

オブジェクト (参照される)

- ・ プリミティブ (実行され変更されない)
- ・ ・ Modifier

クラス

Modifier:

シールされたinterface (Primitive) = 0;

アダプテーション

isEqual

2つのモディファイアはそれらの値が同一である場合にのみ同一である。

【1928】4. 47 ネームド (Named)

Named

クラス

Named:

abstract interface () = (...);

構成

initialize

レスポンドの "name" 属性を新しい割り当てられたテレネーム、現在のプロセスの仲間とする。

【1929】パブリックインスタンス属性

name:

シールされたリードオンリのプロテクトされたテレネーム;

レスポンドの割り当てられたテレネーム。

【1930】4. 48 ニル (Nil)

オブジェクト (参照される)

- ・ プリミティブ (実行され変更されない)
- ・ ・ Nil

クラス

Nil:

シールされたinterface (Primitive) = 0;

アダプテーション

isEqual

いかなる2つの0も同一である。

【1931】4. 49 ナンバ (Number)

オブジェクト (参照される)

- ・ プリミティブ (実行され変更されない)
- ・ ・ Number (順序づけされる)

クラス

Number:

abstract interface (Primitive, Ordered) = (...);

このクラスはシールされている。

【1932】パブリックインスタンスオペレーション

abs:

abstract op () Number;

レスポンドの絶対値をリターンする。

【1933】add:

abstract op (number: Number) Number;

数 ("ナンバ") の算術及びレスポンドをリターンする。

【1934】Ceiling:

abstract op () Integer;

レスポンドよりも算術的に大きいかまたはこれに等しい最小の整数をリターンする。

【1935】divide:

abstract op (divisor: Number) Number

は、DivisionByZeroを送出する。;

レスポンドの算術商、被除数と数 (アーギュメント "divisor")、除数とをリターンする。

【1936】除数が0である場合 ("DivisionByZero") 例外が投出される。

【1937】Floor:

abstract op () Integer;

レスポンドよりも算術的に小さいかまたはこれに等しい最大の整数をリターンする。

【1938】multiply:

abstract op (number: Number) Number;

数 ("number") とレスポンドとの算術積をリターンする。

【1939】negate:

abstract op () Number;

レスポンドの算術ネガティブをリターンする。

【1940】round:

abstract op () Integer;

レスポンドに算術的に最も近い単一の整数またはレスポンドに算術的に最も近い2つの整数の内の1つをリターンする。

【1941】subtract:

abstract op (subtrahend: Number) Number;

被減数であるレスポンドと減数である数（“アギュメント” subtrahend”）との間の算術差をリターンする。

【1942】truncate:

abstract op () Integer;

レスポンドを0に向かって算術的にトランケートするようにレスポンドの分数部分を除くことによって形成された整数をリターンする。

【1943】アダプテーション

isAfter

最初のものが第2のものよりも大きい場合にのみある数は別の数の後になる。

【1944】isBefore

最初のものが第2のものよりも小さい場合にのみある数は別のものの前になる。

【1945】isEqual

2つの数は数学的に等しい場合にのみ相等しい。

【1946】4. 50 オブジェクト (Object)

Object (参照される)

クラス

Object:

abstract interface (Referenced) = (...);

構成

initialize:

プロテクトされないop ();

レスポンド、潜在的なオブジェクト、をイニシャライズしレスポンドをオブジェクトとする。

【1947】finalize:

プロテクトされないop ();

レスポンドがオペレーション”ディスカード”によって破壊されるため要求される。このオペレーションのための方法は、方法のクラスに関してオブジェクトを最終化（ファイナライズ）する。

【1948】パブリックインスタンス属性

Class:

シールされたリードオンリクラス。;

レスポンドがクラス”Class”でなければ、レスポンドが1つのインスタンスであるクラスであり、レスポンドが実際にクラス”Class”であれば、クラス”Class”である。

【1949】size:

シールされたリードオンリ Integer;

オクテットで表したレスポンドの大きさ、すなわち、レ

スポンドの破壊によって再びクレームされるべき近似的な記憶量。

【1950】パブリックインスタンスオペレーション

Copy:

シールされたop () copied Object

は、CopyUnavailableを送出する;

レスポンドへのただ1つの参照を用いてオペレーションがリクエストされた場合にはレスポンドをリターンし、その他の場合にはレスポンドのコピーをリターンする。

【1951】コピーが使用不可能ならば(”CopyUnavailable”)例外が送出される。

【1952】isEqual:

op(object: protected Object) Boolean;

オブジェクト(“アギュメント” object”)がレスポンドに等しいか否かの表示をリターンする。レスポンドまたはオブジェクトが、方法が固有でないクラスのメンバでない場合に、方法はこのオペレーションをエスカレートさせねばならない。クラス”Object”に固有の方法は2つのものを、これらは同一のオブジェクトである場合にのみ等しいとみなす。

select:

シールされたop (associations: Object...

/\* protected Object; Executed \*/

は、Exception, SelectorDuplicatedを送出する。;

オブジェクト及び実行されるオブジェクトの対(“アギュメント” associations”)から、多くとも1つの実行されるオブジェクトを選択し遂行する。選択された1つは、レスポンドのフレーム及びプロパティではなくリクエストのフレーム及びプロパティにアクセスする。

【1953】オブジェクトがレスポンドに等しい場合、そのオブジェクトと対となる実行されるオブジェクトが選択される。その他の場合、オブジェクトが0であれば、そのオブジェクトと対となる実行されるオブジェクトが選択される。その他の場合にはいかなる実行されるオブジェクトも選択されない。

【1954】選択され実行されたオブジェクトがそのようにする場合(”Exception”)または実行されるオブジェクトと対をなす2つのオブジェクトが同等の場合(”SelectorDuplicated”)、例外が投出される。

【1955】インターナルインスタンスオペレーション

getAttribute:

シールされた op (identifier: Identifier)Object  
投出された ClassUnabailable, FeatureUnabailable;  
識別子(“アギュメント” アイデンティファイヤ (identifier)”)によって示されるレスポンドの属性をリターンする。属性は属性について規定された次の通路に従う。

【1956】識別子のクオリファイヤが表すクラスが存在しない場合に(”クラスアンアベイラブル (ClassUnavailable)”)またはリクエストにアクセス可能なレス

ポンドの属性を識別子が表さない場合 (" フィーチャアンアベイラブル (FeatureUnavailble) ") 例外が投出される。

【1957】getClass:

シールドプライベート op (identifier: Identifier!)  
Class

throws ClassUnavailable;

識別子 (" アイデンティファイヤ (identifier) ") が表すクラスを現在の方法にリターンする。クラスが存在しなければ、 (" クラスアンアベイラブル (ClassUnavailable) ") 例外が投出される。

【1958】getProperty:

シールドプライベート op (identifier: Identifier!)  
Object

throws PropertyUndefined;

識別子 (アーギュメント " アイデンティファイヤ (identifier) ") によって表されまた現在の方法がそれに対し固有であるクラスにとって固有なレスポンドのプロパティをリターンする。レスポンドがプロテクトされている場合にのみ、リターンされるリファレンスはプロテクトされている。

【1959】識別子が不適切な時 (" プロパティアンディファインド (PropertyUndefined) ") 例外が投出される。

【1960】setVariable:

シールドプライベート op (identifier: Identifier!)  
Object

throws VariableUndefiend;

識別子 (アーギュメント " アイデンティファイヤ (identifier) ") が表す現在の方法の変数はリターンする。

【1961】識別子が不適切ならば (" バリアブルアンディファインド (VariableUndefiend) ") 例外が投出される。

【1962】setAttribute:

シールドされプロテクトされていない op (identifier: Identifier;  
attribute: Object)

throws ArgumentInvalid, AttributeReadOnly,

ClassUnavailable, FeatureUnavailable;

オブジェクト (アーギュメント " アトリビュート (attribute) ") を、識別子 (アーギュメント " アイデンティファイヤ (identifier) ") によって表されるレスポンドの属性にする。オペレーションは抑制をもしそれが存在すれば最初に廃棄する。オブジェクトは属性について規定された通路に従う。

【1963】オブジェクトが属性のコンストレイントを侵害し (" アーギュメントインヴァリット (ArgumentInvalid) ")、属性がリードオンリであり (" アトリビュートリードオンリ (AttributeReadOnly) ")、識別子のクオリファイヤが表すクラスが存在しない (" クラ

スアンアベイラブル (ClassUnavailable) ") または識別子がリクエストに近接可能なレスポンドの属性を表さない (" フィーチャアンアベイラブル (FeatureUnavailable) ") 場合に例外が投出される。

【1964】setProperty

シールドされプライベートなプロテクトされていない

op (identifier: Identifier!;

property: Object)

throws PropertyUndefined;

オブジェクト (アーギュメント " プロパティ (property) ") を識別子 (アーギュメント " アイデンティファイヤ (identifier) ") によって表され現在の方法がそれに対し固有のクラスに対して固有であるレスポンドのプロパティになる。オペレーションはプロパティが存在すれば最初にそのプロパティを廃棄する。

【1965】識別子が不適切な場合 (" プロパティアンディファインド (PropertyUndefined) ") 例外が投出される。

【1966】setVariable:

シールドされプライベートなプロテクトされていない

op (identifier: Identifier!;

variable: Object)

throws VariableUndefined;

オブジェクト (アーギュメント " ヴァリアブル (variable) ") を識別子 (アーギュメント " アイデンティファイヤ (identifier) ") が表す現在の方法の変数にする。オペレーションは、変数が存在すれば、最初にその変数を廃棄する。

【1967】識別子が不適切であれば (" ヴァリアブルアンディファインド (VariableUndefined) ") 例外が投出される。

【1968】4. 51 オクテット (Octet)

オブジェクト (参照される)

・ プリミティブ (実行され変更されない)

・ ・ Octet (順序あり)

Class

Octet:

シールドされたインタフェース (Primitive, Ordered) =  
0;

アダプテーション

isAfter

あるオクテットは別のオクテットの次にあるのは、それらのものの同様の番号のビットが等しくなく、第1のオクテットの最も高い番号のそのビットが第2のビットの次にある場合にのみ生じる。

【1969】isBefore

あるオクテットは別のオクテットの次にあるのは、それらのオクテットの同じ番号のビットが等しくない、第1のオクテットの最も高い番号のそのようなビットが第2のオクテットの前にある場合に生ずる。

## 【1970】isEqual

2つのオクテットは、どちらも他のものの前にはない場合にのみ相等しい。

## 【1971】Conversions

## Bit

あるビットは、ビット0がそのビットに等しくそのビット1-7がさらに0であるオクテットを生ずる。

## 【1972】Character

そのユニコードのコードが[0, 127]にある整数であるキャラクタは、その整数を変換することによって生ずるオクテットを生ずる。

## 【1973】Integer

[-128, 127]の範囲にある整数は、その整数の2の補数の表示を形成するオクテットを生ずる。ビット7は-128を表す、ビット"n"は、[0, 6]の範囲にある各々の"n"について、 $2^n$ を表す。

## 【1974】OctetString

その1つのアイテムがそのアイテムに等しいアイテムを生ずるオクテット・ストリング。

【1975】4.52 オクテットストリング (Octet String)

オブジェクト (参照される)

- ・ コレクション
- ・ リスト (順序あり)
- ・ コンストレインド・リスト (コンストレインド)
- ・ Octet String (実行)

## Class

OctetString:

シールされたインタフェース (ConstrainedList [Octet], Executed) = (...);

Construction

initialize:

プロテクトされない op (segments: Object...

/\*Octet|protected OctetString! \*/;

レスポンス中のその位置は左から右にかけて増大するセグメントの連鎖 (アークメント "セグメント (segments) ") にする。

## 【1976】アダプテーション

constraint

属性はシールされている。属性の "ofClass", "classId", "isInstance", "isOptional" 及び "passage" の各属性は、クラス "Octet" の識別子 "Octet", "true", "false" 及び "byCopy" である。

## 【1977】isAfter

あるオクテットが別のオクテットの後にあるのは、全て0のオクテットが最初に短い長さのオクテットストリングに付け加えられてその短いオクテットストリングの長さが長いオクテットストリングの長さに等しくなった場合にのみ生ずる。

## 【1978】isBefore

あるオクテットが別のオクテットの前にあるのは、全て0のオクテットが最初に短いオクテット・ストリングに付け加えられてその短い長さのオクテットストリングの長さが長い長さのオクテットストリングに等しくなった場合にのみ生ずる。

## 【1979】Conversions

## Bit

あるビットは、そのビットをオクテットに最初に変換した後オクテットをオクテットストリングに変換することによって生ずるオクテットストリングを生ずる。

## 【1980】BitString

その長さが "n" であるビットストリングは、その長さが "i" であるオクテットストリングを生ずる。"n" は、[8i, 8i+7] の範囲にある。オクテットストリングの位置 "k" にあるオクテットのビット "j" は、 $(8k - j) \leq n$  であれば、オクテットストリングの位置 "k" にあるオクテットのビット "j" は、ビットストリングの位置  $(8k - j)$  にあるビットに等しく、さもなければ0に等しい。

## 【1981】Character

その長さが2であるオクテットストリングを変換することによって生成せうあるキャラクタは、そのオクテットストリングを生ずる。

## 【1982】Integer

整数 "n" (これより小さくないもの) について  $[-2^{8n-1}, 2^{8n-1}]$  の範囲にある整数は、その長さが "n" に等しいオクテットにオクテットストリングに生ずる。2の補数の表示が用いられてその位置が1に等しいオクテットのビットのは  $-2^{8n-1}$  を表し、オクテットストリングの位置 "i" にあるオクテットの1つおきのビット "j" は  $2^{8(n-1)+j}$  を表す。

## 【1983】Octet

オクテットはその唯一つのアイテムがそのオクテットに等しいオクテットストリングを生ずる。

## 【1984】String

あるストリングは、2進テレスク립トにおいてストリングを符号化するための "キャラクターズ (characters) " トークンを生ずる。

## 【1985】4.53 オペレーション (Operation)

オブジェクト (参照される)

- ・ フィーチャ
- ・ Operation

## Class

オペレーション:

シールされたインタフェース (Feature) = (...);

Construction

initialize:

プロテクトされない op (arguments: List [Constraint] | Nil;

```
result: Constraint|Nil;
isPublic: Boolean|Nil;
exceptions: Set [Identifier!]|Nil);
レスポンドの属性を同じ名前のアーギュメント (アーギュメント" アーギュメンツ (arguments) "、" エクセプションズ (exceptions) "、" イズパブリック (isPublic) " 及び" リザルト (result) ") にする。しかしアーギュメント" エクセプションズ (exceptions) " またはアーギュメント" イズパブリック (isPublic) " が 0 であれば対応する属性はクリアされるかまたは、" 偽 (false) " にされる。
```

#### 【1986】Public Instance Attributes

```
arguments:
List [constraint]|Nil;
数が変化しなければ、レスポンドが定義するオペレーションの 0 またはそれ以上のアーギュメントに対するコンストレイントであり、さもなければ 0 である。この場合各々は参照によってパスされマーク以外のどんなオブジェクトでもあり得る。リストの最初のアイテムはスタックの頂部にあるアーギュメントを拘束する。
```

#### 【1987】result:

```
Constraint|Nil;
オペレーションが結果を持てばレスポンドが定義するオペレーションの結果に対するコンストレイントであり、さもなければ 0 である。
```

#### 【1988】アダプテーション

```
isEqual
2つのオペレーションの定義はフィーチャ及びオペレーションの定義に固有のそれぞれの属性に従って固有である。
```

#### 【1989】4. 54 オーダード (Ordered)

```
Ordered
Class
Ordered:
アブストラクトインタフェース () = (...);
パブリックインスタンスオペレーション
isAfter:
アブストラクト op (object:protected Ordered) Boolean;
レスポンドがオブジェクトの後にあるか否かの表示をリターンする (アーギュメント" オブジェクト (object) ") )。
```

【1990】レスポンド及びオブジェクトが、方法がそれに対し固有であるプラスのメンバでない場合に、その方法はこのオペレーションをエスカレーションする。

#### 【1991】isBefore

```
アブストラクト op (Object:protected Ordered) Boolean;
レスポンドがオブジェクトの前にあるか否かの表示をリターンする (アーギュメント" オブジェクト (Object)
```

t) ") )。

【1992】ある方法は、レスポンドまたはオブジェクトが、その方法が固有のクラスのメンバでない場合このオペレーションをエスカレートする。

#### 【1993】max:

```
op (object: Ordered) Ordered;
レスポンドがオブジェクトの後にあれば (アーギュメント" オブジェクト (object) ") レスポンドを、またその他の場合にはオブジェクトをそれぞれリターンする。
```

#### 【1994】min:

```
op (object: Ordered) Ordered;
レスポンドがオブジェクトの前にあれば (アーギュメント" オブジェクト (object) ") レスポンドを、またその他の場合にはオブジェクトをそれぞれリターンする。
```

#### 【1995】4. 55 パッケージ (Package)

オブジェクト (参照される)

- ・ コレクション
- ・ ・ セット (検査される)
- ・ ・ ・ コンストレインド・セット (コンストレインド)
- ・ ・ ・ ・ Package (引用され相互変換される)

Class

Package:

```
インタフェース (ConstrainedSet [Class], Cited, Interchanged) = (...);
```

Construction

initialize:

```
プロテクトされない op (title: Identifier!;
```

```
majorEdition, minorEdition:
```

```
Integer;items: Class...)
```

```
throws ItemDuplicated, ProcessNotPeer;
```

レスポンドのクラス (アーギュメント" アイテムズ (items) ") とし、レスポンドの" サイテーション (citation) " 属性を、その" オーサ (author) " 属性が現在のプロセスの割り当てられたテレネームでありその他の属性が同じ名前のオアーギュメント (アーギュメント" メジャーエディション (majorEdition) "、" マイナーエディション (minorEdition) " 及び" タイトル (title) ") であるサイテーションにする。

【1996】2つのクラスが同一である (" アイテムデュプリケーティッド (ItemDuplicated) ") または現在のプロセスがクラスの仲間でない (" プロセスノットピア (ProcessNotPeer) ") である場合に例外が投出される。

#### 【1997】アダプテーション

constraint

属性はシールされている。属性の" ofClass"、" classId"、" isInstance"、" isOptional" 及び" passage" の各属性は、クラス" Class"、クラス" Class"、" true"、" false"、及び" byCopy" の識別子である。



## 【1998】4. 56 パターン (Pattern)

オブジェクト (参照される)

・ Pattern (Ordered)

Class

Pattern:

インタフェース (object, Ordered) = (…);

Construction

initialize:

プロテクトされていない op (text: copied String!)

throws PatternInvalid;

レスポンドのテキストをストリング (アーギュメント”テキスト (text)”) にする。 提案されたテキストが不適切である場合 (”パターンインヴァリッド (PatternInvalid)”) 例外が投出される。

【1999】パブリックインスタンスオペレーション

find:

op (string: protected String; position: Integer|Nil)

List [Integer] | Nil

throws PositionInvalid

レスポンドにマッチする最長可能な最初のサブストリングについてストリングをサーチする (アーギュメント”ストリング (string)”)。

【2000】このサーチは、整数が提供された場合には、所定の位置 (アーギュメント”位置 (position)”) で、さもなければ、位置1においてそれぞれ開始される。

【2001】オペレーションがマッチを生み出さない場合には0を、その他の場合には2つの整数のリストをそれぞれリターンする。後者の場合第1整数は、マッチ中の最初のキャラクタのストリング中の位置にある。第2整数は、マッチ中の最後のキャラクタのストリング中の位置よりも1だけ大きい。

【2002】提案された位置がそのようなものとして不適切な場合、より詳しくは、0より小さいかまたは0に等しい時 (”ポジションインバリッド (PositionInvalid)”) 例外が投出される。

【2003】注: リスト中の2つの整数はオペレーション”サブストリング (substring)” のアーギュメントとして適切である。

【2004】substitute:

op ( repetitions: Integer;

string: unprotected String!;

replacement: protectedString!)

Integer;

レスポンドにマッチする最長可能なサブストリングについて1の位置で始まる第1ストリング (アーギュメント”ストリング (string)”) をサーチし、第2ストリングのコピーで各々のマッチを取り替えることによって (アーギュメント”リプレースメント (replacemen

t)”) ストリングを変更する。

【2005】オペレーションは、整数 (アーギュメント”レベティションズ (repetitions)”) が0であれば全てのマッチを見出し、さもなければせいぜい同数を見出す。連続するマッチは重なり合わない。1つのマッチが0の長さであれば、次のマッチのサーチは次の位置で始まる。オペレーションは実際に見出されたマッチの数をリターンする。

【2006】代替ストリングは各々の個別の代替に対して合わせることができる。アンドマーク (” &”) が、逆スラッシュ (” \”) を直前に持つ代替ストリング中に表れた場合、代替されるべきサブストリング自身は、アンドマークに変えられ、変更された代替ストリングはサブストリングに変えられる。

【2007】アダプテーション

isEqual

2つのパターンはそのテキストに従って同一である。

【2008】Conversions

String

パターンのテキストとして適切なストリングはそのパターンを作り出す。

【2009】4. 57 パーミット (Permit)

オブジェクト (参照される)

・ Permit (順序あり)

Class

Permit:

インタフェース (オブジェクト、順序あり) = (…);

Construction

initialize:

プロテクトされていない op (age, charges, extent: Integer|Nil);

レスポンドの属性を同じ名前のアーギュメント (アーギュメント”エイジ (age)”, ”チャージズ (charges)” 及び”エクステント (extent)”) とする。レスポンドの1つおきの属性を、属性がブーリアンであれば”真 (true)” にその他の場合は0にする。

【2010】Public Instance Attributes

age:

Integer|Nil;

最大値が課せられた場合レスポンドのサブジェクトの、秒で計った最大のパーミットされるエイジであるか、さもなければ0である。

【2011】authenticity:

Integer|Nil;

最大値が課せられた場合レスポンドのサブジェクトの最大のパーミットされるオーセンティシティか、さもなければ0である。

【2012】charges:

Integer|Nil;

最大値が課せられた場合レスポンドのサブジェクトの、

テーブルリップで計った最大のパーミットされるチャージであるか、さもなければ0である。

【2013】 extent:

Integer|Nil;

最大値が課せられた場合レスポンドのサブジェクトの、オクテットで計った最大のパーミットされるサイズであるか、さもなければ0である。

【2014】 priority:

Integer|Nil;

最大値が課せられた場合レスポンドのサブジェクトの最大のパーミットされるプライオリティであるか、さもなければ0である。

【2015】 canCharge:

Boolean;

オペレーション” charge”を要求するのにレスポンドのサブジェクトがパーミットされるか否か。

【2016】 canCreate:

Boolean;

クラス”サブクラス (subclass)” のオペレーション” new”を要求する上にレスポンドのサブジェクトがパーミットされるか否か。

【2017】 canDeny:

Boolean;

仲間のプロセスの能力を減少させるように仲間のプロセスの属性”ネイティブパーミット (nativepermit)” をセットする上にレスポンドのサブジェクトがパーミットされるか否か。

【2018】それと一致して、他の仕方でするようにする資格が与えられている場合、レスポンドのサブジェクトが、プロセス”リージョナルパーミット (regionalPermit)” または”ローカルパーミット (localPermit)” の各属性をセットするようにパーミットされているか否か。

【2019】 canGo:

Boolean;

レスポンドのサブジェクトがオペレーション” go” をリクエストするようにパーミットされているか否か。

【2020】 canGrant:

Boolean;

仲間のプロセスの能力を増大させるように仲間のプロセスの属性”ネイティブパーミット (nativePermit)” をセットする上にレスポンドのサブジェクトがパーミットされているか否か。

【2021】それと一致して、他の仕方でするようにする資格が与えられている場合、レスポンドのサブジェクトが、プロセス”リージョナルパーミット (regionalPermit)” または”ローカルパーミット (localPermit)” の各属性をセットするようにパーミットされているか否か。

【2022】 canRestart:

Boolean;

レスポンドのサブジェクトが再スタートするようにパーミットされているか否か。

【2023】 canSend:

Boolean;

レスポンドのサブジェクトはオペレーション” send” をリクエストするようにパーミットされているか否か。

【2024】 Public Instance Operations

intersections

op (permit: Permit) Permit;

アーギュメント (アーギュメント”パーミット (permit)”) とレスポンドとの交点をリターンする。

【2025】 アダプテーション

isAfter

1つのパーミットが次のパーミットの後になるのは、これらのパーミットが同一でなく第1のパーミットの固有の属性が第2のパーミットのそれの前にある、第1のパーミットの少なくとも1つの固有の属性が第2のものの属性の後にある場合のみである。

【2026】 isBefore

1つのパーミットが次のパーミットの前にあるのは、これらのパーミットが同一でなく第1のパーミットの本来の属性が第2のパーミットのその後にある、第1のパーミットの少なくとも1つの固有の属性が第2のパーミットの固有の属性の前にある場合のみである。

【2027】 isEqual

2つのパーミットはそれらのものの固有の属性に従って同一である。

【2028】 4. 58 ペティション (Petition)

オブジェクト (参照される)

・ Petition

Class

Petition:

インタフェース = (...);

Construction

initialize:

プロテクトされていない op (agentName: Telename|Nil;

agentClass: Citation|Nil;

maximumWait: Integer|Nil);

レスポンドの本来の属性を同じ名前のアーギュメント

(アーギュメント” エージェントラス (agentClass) ”、” エージェントネーム (agentName) ” 及び” マキシマムウェイト (maximumWait) ”) とする。

【2029】 Public Instance Attribute

agentClass

Citation|Nil;

もしそのようなコンストレイントが課された場合、レスポンドが定義するミーティングのペティションニーがメン

バであるクラス” エージェント (agente) ” のサブクラスであるクラスのサイテーションであるか、さもなければ0である。

【2030】 agentName:

Telename|Nil;

そのようなコンストレイントが課せられるならば、レスポンドが定義するミーティングのペティションのテレネームであるか、さもなければ0である。

【2031】 maximumWait:

integer|Nil;

そのようなコンストレイントが課せられるならば、レスポンドが定義するミーティングが取り決められる時とミーティングが取り決められる時との量で表された最大の許容差が、さもなければ0である。供給される場合の整数は負数ではない。

【2032】 アダプテーション

isEqual

2つのペティションは固有の属性に従って同一である。

【2033】 4. 59 ペティションド (Petitioned)

Petitioned

Class

Petitioned:

アブストラクトインタフェース 0 = (...);

System Instance Operations

meeting:

プロテクトされない op (contact: Contact;

petition: protected Petition)

throws MeetingDenied;

コンタクト (アーギュメント” コンタクト (contact) ”) のサブジェクトがペティション (アーギュメント” ペティション (petition) ”) を用いてレスポンドに合う前に成功の内に遂行される。オペレーションが遂行されている間、コンタクトのサブジェクトの属性は0であり、オペレーションが成功した場合にのみ” サブジェクト (subject) ” にされる。

【2034】 ミーティングは拒絶 (“ ミーティングディナイド (MeetingDenied) ”) された場合には例外が投出される。クラスに固有のこのオペレーションの方法は例外を投出する。

【2035】 parting:

プロテクトされない op (contact: Contact);

コンタクト (アーギュメント” コンタクト (contact) ”) のサブジェクトがレスポンドに合った道にリクエストされる。エンジンは、オペレーションがリクエストされる前に、コンタクトの” サブジェクト (subject) ” を0にする。クラス” ペティションド (Petitioned) ” に固有の方法は効果を持たない。

【2036】 4. 60 プレイス

オブジェクト (参照される)

・ プロセス (名前付けされる)

・ ・ Place (移動されない)

Class

Place:

アブストラクトインタフェース (Process, Unmoved) = (...);

Construction

initialize

レスポンドの” アドレス (address) ” 属性を現在のプロセスの仲間である新しく割り当てられたテレネームとし、レスポンドの属性” パブリッククラス (publicClasses) ” をクリアする。

【2037】 Public Instance Attributes

address:

シールされたリードオンリのプロテクトされたテレアドレス;

レスポンドの割り当てられたテレアドレス

publicClasses:

シールされたリードオンリのセット [引用される];

レスポンドがここでその占有者に近接可能にするクラス、パッケージまたは両方のセット。属性は、リクエストがプレイスである場合に、プロテクトされないリファレンスである、さもなければプロテクトされたリファレンスである。

【2038】 パブリックインスタンスオペレーション

terminate:

シールされたプロテクトされていない op

(占有者: protected Telename) Boolean

throws ProcessNotCurrent, ProcessNotPeer,

StateImproper;

割り当てられたテレネーム (アーギュメント” オキュパント (occupant) ”) を有するレスポンドの占有者を終了させ、そのようなプロセスが合ったか否かの指示をリターンする。プロセスはプロセスの固有のパーミットを強制的に使い果すことによって終了する。

【2039】 レスポンドが現在のプレイスでない場合 (“ プロセスノットカレント (ProcessNotCurrent) ”) 、現在のプロセスが占有者の仲間でない (“ プロセスノットピア (ProcessNotPeer) ”) またはリクエストの状態がオペレーション” ターミネート (terminate) ” を排除する場合 (“ ステートインプロバ (StateImproper) ”) 例外が投出される。

【2040】 システムインスタンスオペレーション

entering:

プロテクトされない op (contact: Contact;

permit: protected Permit;

ticket: protected Ticket|Nil)

throws OccupancyDenied;

コンタクト (アーギュメント” コンタクト (contact) ”) のサブジェクトがレスポンドを占有する前に成功の内に遂行される。コンタクトの” サブジェクト (subject) ”

bject) ” の属性は、オペレーションがリクエストされる場合には0であるが、オペレーションが成功した場合にのみサブジェクトとされる。

【2041】サブジェクトは、チケットとともに到着するエージェントが供給されれば、そのようなエージェント（アージェント” チケット (ticket) ”）であり、0が供給されれば、レスポンドにおいて作り出されているプロセスである。

【2042】パーミット（アージェント” パーミット (permit) ”）はサブジェクトの提案された最初のローカルパーミットである。

【2043】このオペレーションが成功した後、レスポンドがコンタクトされるオブジェクトである場合にのみ、エンジンはレスポンドの” コンタクトされた (Contacted) ” 属性中にコンタクトを含める。

【2044】コンタクトのサブジェクトが占有を拒絶する場合（” オキュパンシディナイド (OccupancyDenied) ”）にのみ例外が投出される。このクラスに固有のオペレーションの方法は例外を投出させる。

【2045】注：チケットが供給された場合、パーミットは同一であり、従ってチケットの属性” デスティネーションパーミット (destinationPermit) ” を複製する。

【2046】exiting:

プロテクトされていない op (contact: Contact;

permit: protected Permit;

ticket: protected Ticket|Nil);

コンタクトのサブジェクト（アージェント” コンタクト (contact) ”）は、1度リクエストされたら、レスポンドをもはや占有しない。コンタクトの” サブジェクト (subject) ” の属性は、オペレーションが要求された時に0である。クラスの固有のオペレーションは効果がない。

【2047】サブジェクトは、チケットとともにレスポンドをさるエージェントを（アージェント” サブジェクト (subject) ”）は供給された場合エージェントであり、エージェントが供給された場合、レスポンド中において破壊されるプロセスである。パーミット（アージェント” パーミット (permit) ”）は、サブジェクトの現在のローカルパーミットである。

【2048】このオペレーションが遂行された後、レスポンドがコンタクトされるオブジェクトである場合、エンジンは、レスポンドの” contacted ” 属性からコンタクトを除外する。

【2049】4. 6 1 プリミティブ (Primitive)

オブジェクト (参照される)

・ Primitive (実行され変更されない)

Class

Primitive:

アブストラクトインタフェース (オブジェクト、実行さ

れ、変更されない) = (...);

このクラスはシールされる。

【2050】Construction

initialize:

プロテクトされていない op 0

throws FeatureUnavailable;

例外を投出する（” フィーチャアンアベイラブル (FeatureUnavailable) ”）。

【2051】4. 6 2 プリミティブエクセプション (Primitive Exception)

オブジェクト (参照される)

・ 例外 (参照される)

・ ・ プログラミングの例外

・ ・ ・ Primitive Exception

Class

PrimitiveException:

アブストラクトインタフェース (プログラミングの例外) = 0;

Subclasses

DivisionByZero

インタフェース (PrimitiveException) = 0;

除数は0である。

【2052】4. 6 3 プロシージャ (Procedure)

オブジェクト (参照される)

・ プリミティブ (実行され変更されない)

・ ・ Procedure

Classs

Procedure:

シールドインタフェース (プリミティブ) = (...);

Adaptations

isEqual

2つプロシージャは長さが等しくその同じ位置にあるアイテムが同一である場合にのみ同一である。

【2053】Conversions

List

そのアイテムが実行されるオブジェクトのリストは、同じ長さのプロシージャを作り出し、このプロシージャの各々のアイテムは、同じ位置にあるもののコピーである。

【2054】OctetString

2進テレスクリプトであるオクテットストリングは、そのアイテムが2進テレスクリプトによって符号化されるオブジェクトであるプロシージャを作り出す。

【2055】String

キャラクタテレスクリプトであるストリングは、そのアイテムがキャラクタ・テレスクリプトによって符号化されるオブジェクトであるプロシージャを作り出す。

【2056】4. 6 4 プロセス (Process)

オブジェクト (リファレンスされる)

・ Process (Named)

Class

Process:

アブストラクトインタフェース (Object, Named, Unco-  
ied) = (...);

このクラスはシールされる。

【2057】Construction

initialize:

op ( nativePermit: copied Permit;

privateClasses: Set [Cited] [Nil]

throws PermitViolated;

レスポンドの属性”ネーム (name)” を、現在のプロセスの仲間であり、新しく割り当てられたテレネームズ、レスポンドの属性を、同じ名前前のアーギュメント (アーギュメント”ネイティブパーミット (nativePermit)”)、レスポンドの属性”プライベートクラシズ (privateClasses)” を、アーギュメントが0でない場合に、同じ名前前のアーギュメント (アーギュメント”プライベートクラシズ (privateClasses)”) とし、さもなければ、現在のプロセスの同じように特定された属性とし、レスポンドの属性コンタクトを、クリアされたセットとする。

【2058】現在のプロセスの内有効なパーミットがオペレーション”permit”を禁止し、そのパーミットはその他の点で不適切であり、またはオペレーション”entering”が失敗 (“パーミットヴァイオレイティッド (PermitViolated)”) した場合に、例外が投出される。

【2059】注: クラス”プロセス”のサブクラスに固有のオペレーションの方法は、作り出されたものでなく、作り出すプロセスを現在のプロセスと見る。

【2060】Public Instance Attributes

brand:

シールされたリードオンリのプロテクトされたオブジェクトはプロセスノットピアを投出させる。

【2061】レスポンドのブランド。

【2062】現在のプロセスがレスポンドの仲間でない場合 (“プロセスノットピア (ProcessNotPeer)”) 例外が投出される。

【2063】localPermit:

シールされコピーされたパーミットは、FeatureUnavailable, PermitViolatedを投出する。

【2064】レスポンドのローカルパーミット

リクエストがレスポンドでもなくレスポンドが占有する場合 (“フォーチャアンアベイラブル (FeatureUnavailable)”) またはプロセスモデルと合致しない仕方では属性を設定する試みがなされた場合 (“パーミットヴァイオレイティッド (PermitViolated)”) 例外が投出される。

【2065】nativePermit:

シールされコピーされたパーミットは、FeatureUnavailable, PermitViolatedを投出する。

【2066】レスポンドの本来のパーミット。

【2067】リクエストがレスポンドでもその仲間でもない (“フィーチャアンアベイラブル (FeatureUnavailable)”) またはプロセスモデルと合致しない仕方では属性を設定する試みがなされた場合 (“パーミットヴァイオレイティッド (PermitViolated)”) 例外が投出する。

【2068】permit

シールされたリードオンリのコピーされたパーミット。

【2069】レスポンドの有効なパーミット。

【2070】regionalPermit

シールされたコピーされたパーミットは、FeatureUnavailable, PermitViolatedを投出する。

【2071】レスポンドの地域的なパーミット。

【2072】リクエストがレスポンドでもエンジンプレイスでもない (“フィーチャアンアベイラブル (FeatureUnavailable)”) またはプロセスモデルと合致しない仕方では属性を設定する試みがなされた場合 (“パーミットヴァイオレイティッド (PermitViolated)”) 例外が投出する。

【2073】privateClasses:

シールされたリードオンリのセット [引用される] は、ProcessNotPeerを投出する。

【2074】レスポンドが保持するクラス、パッケージまたは両方のセット。

【2075】現在のプロセスがレスポンドの仲間でない場合 (“プロセスノットピア (ProcessNotPeer)”) 例外が投出される。

【2076】パブリックインスタンスオペレーション charge:

シールされた op (charges: Integer) は、PermitInadequate, PermitViolatedを投出する。

【2077】整数の量によってレスポンドの実際のチャージを増大させる (アーギュメント”チャージズ (charges)”)。

【2078】レスポンドの有効なパーミットが使い尽くされる (“パーミットインアデクェット (PermitInadequate)”) かまたは現在のパーミットがオペレーション”チャージ”を禁止する (“パーミットヴァイオレイティッド (PermitViolated)”) 場合、例外が投出される。

【2079】restrict:

シールされた op (procedure: Procedure; permit: protected Permit)

は、Exception, PermitViolated, ProcessNotCurrentを投出する。

一時的なパーミット (アーギュメント”パーミット (permit)”) 及びプロシージャのチャージが生ずる現在のプロセスの結果する実効的なパーミットのもとに、プロシージャ、(アーギュメント”プロシージャ (procedur

e) ”) を遂行する。

【2080】プロシージャがそのようにする場合(“例外(exception)” )、プロシージャの遂行が前述した実効的なパーミットを侵害する場合(“パーミットバイオレーティッド(PermitViolatd)” ) またはリクエストが現在のプロセスでない場合(“プロセスノットカレント(ProcessNotCurrent)” ) 例外が投出される。

【2081】sponsor

シールされた op (procedure: Procedure; permit: protected

Permit)は、Exception, PermitViolatd, ProcessNotCurrentを投出する。

一時的なパーミット(“パーミット(permit)” ) 及びプロシージャのチャージが生ずるレスポンドのオーナーの結果した実効的パーミットのもとに、プロシージャ(“アーギュメント” プロシージャ(procedure)” ) を遂行する。プロシージャはレスポンドのフレーム及びプロパティでなくリクエストのフレーム及びプロパティにアクセスする。

【2082】プロシージャがそのようにする場合(“例外(Exception)” )、プロシージャの遂行が前述した実効的なパーミットを侵害している場合(“パーミットバイオレーティッド(PermitViolatd)” ) またはリクエストが現在のプロセスでない場合(“プロセスノットカレント(ProcessNotCurrent)” ) 例外が投出される。

【2083】Private Instance Attributes

age:

シールされたリードオンリの整数。

【2084】レスポンドの秒で表した実際のエイジ。

【2085】charges:

シールされたリードオンリの整数。

【2086】レスポンドの実際のテレクリックで示したチャージ。

【2087】priority:

シールドInteger|Nil;

レスポンドの実際のプライオリティすなわちレスポンドの選択されたプライオリティ。

【2088】プライベートインスタンスオペレーション wait

プロテクトされない op (seconds: Integer);

要求された秒数(“アーギュメント” セカンズ(second s)” ) が経過するまでレスポンドをブロックする。整数が負数である場合にはオペレーションは効果を持たない。

【2089】システムインスタンスオペレーション live:

アブストラクト、プロテクトされない op (cause: Exception|Nil)は、Exceptionを投出する。

生成させた後レスポンドを開始させるために0でリクエストする(“アーギュメント” コーズ(cause)” )。レスポンドが保持するパーミットの属性” キャンリスタート(canRestart)” が” 真(true)” であれば、そのオペレーションは、レスポンドが例外(“アーギュメント” コーズ(cause)” ) によって失敗した場合に、レスポンドを再スタートさせる前記例外とともにリクエストされる。

【2090】レスポンドが例外をキャッチしえなかった場合(“例外(Exception)” ) 例外が投出される。

【2091】restricted

op (permit: Identifier; isRelocated: oolean)

PermitReduced|Nil;

レスポンドの属性” nativePermit”、” regionalPermit” または” localPermit” が一度設定された後にレスポンドの能力を減少させるように再び設定された場合にリクエストされる。識別子(“アーギュメント” パーミット(permit)” ) はこれらの3つの属性の内どれか設定されたものの識別子である。結果は0でない場合にエンジンはレスポンドのスレッドにオペレーションの結果を投出する。このクラスにおいて固有の方法は単に0をリターンする。

【2092】プーリアン(“アーギュメント” イズリロケートィッド(isRelocated)” ) は、オペレーションをリクエストするようにエンジンを導いた事象がレスポンドが占有しているプレイスまたは、オペレーション” go” または” send” の遂行の間にオペレーションがリクエストされた場合にはレスポンドの目的点以外のプレイス例えばパーガトリィにレスポンドを送り届けるようにエンジンを導いたか否かを指示する。後者の場合オペレーション” go” または” send” は、0でない場合に、現在のオペレーションの結果を投出する。

【2093】注: レスポンドの新しく設定されたパーミットが著しく制限的である場合に、このオペレーションを遂行するレスポンドの能力は、プロセスの終了のルールによって制限される。

【2094】アダプテーション

copy

” コピーアンアベイラブル(Copy Unavailable)” のメンバが投出される。

【2095】4. 65 プロセスエクセプション(Process Exception)

オブジェクト(参照される)

- ・ エクセプション(変更されない)
- ・ ・ プログラミングの例外
- ・ ・ ・ Process Exception

Class

ProcessException:

アブストラクトインタフェース(プログラミングの例

外) = 0;

Subclasses

PermitInadequate:

インタフェース (プロセスの例外) = 0;

あるプロセスは、別のプロセスに、この別のプロセスの実効的なパーミットを使い尽くすようにチャージするように試みる。

【2096】ConditionUnavailable:

インタフェース (プロセスの例外) = 0;

あるプロセスは、リソースを専用的に使用することなくリソースの状態をマニピュレートする。

【2097】ConditionUndefined:

インタフェース (プロセスの例外) = 0;

リソースの状態を表すように意図される識別子はそのようにしない。

【2098】PermitExhausted:

インタフェース (プロセスの例外) = 0;

あるプロセスはプロセスが保持するパーミットを使い尽くす。

【2099】PermitViolated:

インタフェース (プロセスの例外) = 0;

あるプロセスはそれが保持するパーミットを侵害する。

【2100】ProcessNotCurrent:

インタフェース (プロセスの例外) = 0;

あるプロセスは、あるクラスの現在のメンバのフィーチャをリクエストする。

【2101】ProcessNotPeer:

インタフェース (プロセスの例外) = 0;

あるオブジェクトは、そのオブジェクトの仲間でないプロセスをマニピュレートする。

【2102】ResourceUnavailable:

インタフェース (プロセスの例外) = 0;

リソースの使用は、要求された秒の数において取得できない。

【2103】StateImproper

インタフェース (プロセスの例外) = 0;

プロセスは、リクエストを排除する状態にある間、あるフィーチャをリクエストする。

【2104】4. 66 プログラミングエクセプション (ProgrammingException)

オブジェクト (参照される)

・ 例外 (変更されない)

・ ・ Programming Exception

Class

ProgrammingException:

アブストラクト・インタフェース (例外) = 0;

4. 67 クオリファイドアイデンティファイヤ (Qualified Identifier)

オブジェクト (参照される)

・ プリミティブ (実効され、変更されない)

・ ・ 識別子 (順序あり)

・ ・ ・ Qualified Identifier

Class

QualifiedIdentifier:

シールされたインタフェース (識別子) = 0;

4. 68 ランダムストリーム (RandomStream)

オブジェクト (参照される)

・ ストリーム

・ ・ Random Stream

Class

RandomStream:

インタフェース (Stream [Integer]) = (…);

Construction

initialize:

プロテクトされていない op (seed: Integer)

throws SeedInvalid;

レスポンドのシードを整数 (アーギュメント”シード

(seed)”) とし、レスポンドの属性”カレント (curr

ent) ” を 0 とし、レスポンドの属性”ネクスト (nex

t) ” を定義されないままにしておく。

【2105】提案されたシードがそのようなものとして不適切な場合 (“SeedInvalid”) 例外が投出される。

【2106】アダプテーション

current

この属性は整数である。

next

この属性は 0 ではなく整数である。

【2107】4. 69 実数 (Real)

オブジェクト (参照される)

・ プリミティブ (実効され、変更されない)

・ ・ 数 (順序あり)

・ ・ ・ Real

Class

Real:

シールされたインタフェース (数) = 0;

アダプテーション

abs

結果は実数である。

【2108】add

結果は実数である。

divide

結果は実数である。

multiply

結果は実数である。

negate

結果は実数である。

subtract

結果は実数である。

Conversions

Bit

あるビットは、そのビットを整数に最初に変換し次にその実数を整数に変換する結果を生ずる。

#### 【2109】BitString

あるビットストリングは、そのビットストリングを最初に整数に変換し次にその整数を実数に変換する結果を生ずる。

#### 【2110】Boolean

ブーリアンは、そのブーリアンを整数に最初に変換し次にその整数を実数に変換する結果を生ずる。

#### 【2111】Character

あるキャラクタは、そのキャラクタを整数に最初に変換し次にその整数を実数に変換する結果を生ずる。

#### 【2112】Integer

整数は、算術的にその整数に等しい実数を生ずる。

#### 【2113】Octet

オクテットは、そのオクテットを整数に最初に変換し次にその整数を実数に変換する結果を生ずる。

#### 【2114】OctetString

オクテット・ストリングは、そのオクテット・ストリングを整数に最初に変換し次にその整数を実数に変換する結果を生ずる。

#### 【2115】String

キャラクタテレスクリプトの実数のトークンのシンタックスルールに従うルールは、対応する実数を生ずる。

#### 【2116】4. 70 リファレンスド (Referenced)

Referenced

Class

Referenced:

アブストラクトインタフェース 0 = (...);

このクラスはシールされる。

#### 【2117】Public Instance Attributes

isProtected

シールされたリードオンリのブーリアン;

レスポンドのリファレンスがプロテクトされているかどうか。

#### 【2118】パブリックインスタンスオペレーション

discard:

シールド op 0;

レスポンドへのリファレンスが全く残っていない場合にのみレスポンドを破壊する。

#### 【2119】isSame:

シールされた op (reference: Protected Referenced) Boolean;.

リファレンスされたオブジェクト (アーギュメント"リファレンス (reference)" ) がレスポンドであるかどうかの指示をリターンする。

#### 【2120】protect:

シールされた op 0 protected Reference;

プロテクトされたリファレンスをレスポンドにリターンする。

#### 【2121】ref:

シールされプロテクトされていない op 0;

オペレーションをリクエストするために用いられたリファレンスそれ自体が保護されている場合にのみともにプロテクトされるレスポンドへの2つのリファレンスをスタックに保存する。

#### 【2122】4. 71 リソース (Resource)

オブジェクト (参照される)

・ Resource

Class

Resource:

インタフェース = (...);

Constructin

initialize:

プロテクトされない op (conditon: Identifier!Nil;

conditons:copied Set[Identifier!Nil])

throws ConditionUndefined;

レスポンドの固有の属性を、同じ名前のアーギュメント (アーギュメント"コンディション (conditon)" 及び"コンディションズ (conditons)" ) とする。どちらかのアーギュメントが0である場合、両方とも0でなくてはならず、その場合に、属性"コンディションズ (conditons)" は、属性"コンディション (conditon)" に等しい1つの識別子を含む。問題の識別子は定義されない。

【2123】個別の条件すなわち属性"条件" がセットにない場合、すなわち属性"コンディションズ" にない場合 ("コンディションアンディファインド (ConditionUndefined)" ) 例外が投出される。

#### 【2124】Public Instance Attributes

condition:

識別子

throws ConditionUnavailable;

レスポンドの条件の識別子。

【2125】現在のプロセスが属性を設定しようと試み、レスポンドの排他的な使用が欠如している場合 ("コンディションアンアベイラブル (ConditionUnavailable)" ) には例外が投出される。

#### 【2126】conditions:

リードオンリのプロテクトされるセット [識別子]

レスポンドの可能な条件のための識別子。

#### 【2127】パブリックインスタンスオペレーション

use:

シールされプロテクトされない op (procedure: Procedure;

exclusive: Boolean|Nil;

maximumWait: Integer|Nil;

conditions: copied Set [Identifier!Nil])

ブーリアン

ConditionUndefined, Exception, resourceUnavailable



を投出する。

【2128】レスポンドの使用を取得する、プロシージャ（アークメント”プロシージャ（procedure）”）を遂行し次にプロシージャが失敗してもレスポンドを放棄する。

【2129】プロシージャはレスポンドのでなくリクエストのフレーム及びプロパティにアクセスする。

【2130】ブーリアン（アークメント”エクスクルーシブ（exclusive）”）が存在し且つ”真”である場合にのみ使用が排他的となる。もし0でなく、識別子のセット（アークメント”コンディションズ（conditions）”）が供給されたら、レスポンドが識別子が意味する状態の1つにある場合にのみ使用が許可される。

【2131】整数（アークメント”マキシマムウェイト（maximumWait）”）が供給されたらオペレーションは、その秒数よりも多くない時間レスポンドの使用に対して待機する。その代わり0が供給されたら、オペレーションは要求されるだけの時間、おそらくは永久に待機する。オペレーションの結果は、前者の場合に、指示された秒数を持ちレスポンドが利用できないままであるか否かを指示する。

【2132】条件のセットがクリアされるかまたはレスポンドの属性”コンディション”（”コンディションアンディファインド（ConditionUndefined）”）のサブセットに等しくない場合、プロシージャが例外を投出する場合（”エクセプション（Exception）”）、またはレスポンドの使用が要求された秒数の間に取得できない場合（”リソースアンアベイラブル（ResourceUnavailable）”）例外が投出される。

【2133】4. 72 セレクタ（Selector）  
オブジェクト（参照される）

- ・ プリミティブ（実行され変更されない）
- ・ ・ Selector

Class

Selector:

シールされたインタフェース（プリミティブ） = 0;

アダプテーション

isEqual

2つの値が同一である場合にのみ2つのセレクタは同一である。

【2134】4. 73 セット（Set）

オブジェクト（参照される）

- ・ コレクション
- ・ ・ Set（検査される）

Class

Set:

インタフェース [itemClass: Class]

(Collection[itemClass, Verifid] = (...));

サブジェクトクラスの各々のメンバの各々のアイテムがそれ自身メンバであるクラス（アークメント”アイテ

ムクラス（itemClass）”）によってパラメータ化される。

【2135】レスポンドのアイテムをオブジェクトとする（アークメント”アイテムズ（items）”）。

【2136】2つの提案されたアイテムが同一である場合（”アイテムデュプリケートッド（ItemDuplicated）”）または提案されたアイテムがそのようなものとして不適切である場合（”アイテムインヴァリッド（ItemInvalid）”）に例外が投出される。

【2137】パブリックインスタンスオペレーション difference:

プロテクトされない op (set: protected Set [itemClass]);

セットの1つのアイテムに等しい各々のアイテム（アークメント”セット（set）”）をレスポンドから除外して廃棄する。

【2138】intersection:

プロテクトされない op (set: protected Set [itemClass]);

セットの1つのアイテムに等しくない各々のアイテム（アークメント”セット（set）”）をレスポンドから除去し廃棄する。

【2139】union:

プロテクトされない op (set: protected Set [itemClass]);

レスポンドの現存するアイテムに等しくないセット（アークメント”セット（set）”）の各々のアイテムへの参照をレスポンドに含める。

【2140】アダプテーション

include

オブジェクトに等しい現存するアイテムを最初に排除し次に廃棄することによって、あるオブジェクトを新しいアイテムとしてセットに含める。

【2141】verify

セットのアイテムどの2つも同一でない場合にのみセットはコンシステントである。

【2142】4. 74 スタック（Stack）

オブジェクト（参照される）

- ・ コレクション
- ・ ・ リスト（順序付けされる）
- ・ ・ ・ Stack

Class

Stack:

インタフェース [itemClass: Class] (List [item]) = (...);

サブジェクト・クラスの各々のメンバの各々のアイテムそれ自身がメンバであるクラス（アークメント”アイテムクラス（itemClass）”）によってパラメータ化される。

【2143】パブリックインスタンスオペレーション

pop:

プロテクトされない op 0 itemClass

throws StackDepleted;

トップアイテムをレスポンドから除去しリターンする。

【2144】レスポンドがクリアされている場合 (" スタックデプリーティッド (StackDepleted) ") に例外が投出される。

【2145】push:

プロテクトされない op (item: itemClass);

オブジェクト (アーギュメント" アイテム (item) ") を新しいアイテムとして頂部においてレスポンドに付加する。

【2146】pushItems:

プロテクトされない op (item: protected List [itemClass]);

リストのアイテム (アーギュメント" アイテムズ (items) ") への参照をレスポンドにプッシュする。リストの位置1においてアイテムへのリファレンスをレスポンドの頂部に置く。

【2147】roll:

プロテクトされない op (shifts, items: Integer)

throws ArgumentInvalid, StackDepleted;

レスポンドの一番上のアイテムをシフトする。

【2148】シフトに参加するアイテムの数 "I" は、負数でない整数 (アーギュメント" アイテムズ (items) ") によって与えられる。各々の参加するアイテムがシフトされるべき位置の数は、整数でも負数でもよい別の整数 (アーギュメント" シフツ (shifts) ") によって与えられる。" P" はその絶対値である。参加するアイテムは、この第2の整数が正数である場合レスポンドの頂部にシフトされ、さもなければレスポンドの底部にシフトされる。

【2149】レスポンドの頂部に向かって位置1だけ参加するアイテムをシフトさせることは、1番最上部のアイテムの位置を "アイテム" に変化させ他の参加する各々のアイテムを位置を1だけ減少させることである。

【2150】レスポンドの底部に向かって参加アイテムを位置1だけシフトさせることは、その位置が "I" であるアイテムを1番上にし、他の参加アイテムの各々の位置を1だけ増大させることである。

【2151】"I" を与える正数が負である場合 ("アーギュメントインヴァリッド (ArgumentInvalid) ") またはレスポンドの長さが "I" (" スタックデプリーティッド (StackDepleted) ") よりも小さい場合に例外が投出される。

【2152】swap:

プロテクトされない op 0 は、

StackDepleted; を投出する。

【2153】レスポンドの位置1及び2にあるアイテムを転置する。

【2154】レスポンドの長さが2より小さい場合 (" スタックデプリーティッド (StackDepleted) ") 例外が投出される。

【2155】4. 75 ストリーム (Stream)

オブジェクト (参照される)

・ Stream

Class

Stream:

アブストラクトインタフェース [itemClass: Class] = (···);

サブジェクトクラスの各々のメンバの各々のアイテムがそれ自身メンバであるクラス (アーギュメント" アイテムクラス (itemClass) ") によってパラメータ化される。

【2156】パブリックインスタンスオペレーション

current:

アブストラクトのリードオンリの itemClass|Nil;

レスポンドの属性 "ネクスト (next) " が質疑された

が、0がリターンされず、全てのレスポンドのアイテムが作り出された場合には、レスポンドが最後に作り出したアイテムであり、さもなければ0である。

【2157】isDone:

アブストラクトなブーリアン。

【2158】レスポンドが全てのアイテムを作り出したか否か。

【2159】next:

アブストラクトなリードオンリの itemClass|Nil

throws ReferenceProtected;

もしレスポンドがそのアイテムの全てをすでに作り出していた場合には、レスポンドの次のアイテムである、さもなければ0である。このオペレーションがリクエストされるごとにレスポンドは別のアイテムを作り出す。

【2160】レスポンドがプロテクトされている場合 (" リファレンスプロテクティッド (Referenceprotected) ") 例外が投出される。

【2161】4. 76 ストリング (string)

オブジェクト (参照される)

・ コレクション

・ ・ リスト (順序付けされる)

・ ・ ・ コンストレインドリスト (コンストレインド)

・ ・ ・ ・ String (Cased&Executed)

Class

String:

シールされたインタフェース

(ConstrainedList [Character], Cased, Executed) = (···);

Construction

initialize

プロテクトされない op (segments: Object···

/\*Character|protected String! \*/);

レスポンドをその位置がレスポンド内において左から右に増大しているセグメント（“アークメント”セグメント（segments）”）の連鎖にする。

【2162】パブリックインスタンスオペレーション  
substring:  
op (initialPosition, beyondFinalPosition: Integer)  
copied String  
throws PositionInvalid;

範囲に含まれる位置においてキャラクタを含むレスポンドのサブstringのコピーをリターンする [“initialPosition”, “beyondFinalPosition”]。

【2163】主張された位置がそのようなものとして不適切な場合 (“ポジションインヴァリッド (PositionInvalid)”) 例外が投出される。

【2164】アダプテーション  
constraint

属性はシールされる。属性 “ofClass”, “classId”, “isInstance”, “isOptional” 及び “passage” は、クラス “Character”, クラス “character”, “true”, “false” 及び “byCopy” の識別子である。

【2165】isAfter  
短い長さのstringの長さは長い長さのstringの長さに等しくするに足る文字キャラクタをキャラクタが最初に短な長さのstringに付加されたかのようにあるstringが次のstringに続く。各々の付加されたキャラクタは、そのユニコードのコードが正数0であるキャラクタである。

【2166】isBefore  
短い長さのstringの長さは長い長さのstringの長さに等しくするに足るキャラクタをキャラクタが最初に短い長さのstringに付加されたかのように、あるstringが別のstringの前方にある。各々の付加されたキャラクタは、そのユニコードのコードが正数0であるキャラクタである。

【2167】Conversions  
Bit

あるstringに変換しうるキャラクタに変換しうるビットはそのstringを生ずる。

【2168】CalendarTime  
カレンダー時間は、カレンダー時間によって表される時間を記述するstringを生ずる。

【2169】Character  
キャラクタは、その唯一つのアイテムがキャラクタに等しいstringを生ずる。

【2170】Identifier  
識別子は、識別子のテキストに等しいstringを生ずる。

【2171】Integer  
ある整数は、キャラクタテレクリプト中のトークン “インテジャ (Integer)” のためのシンタクティックル

ールに従うstringを生ずる。

【2172】Pattern  
あるパターンは、パターンのテキストに等しいstringを生ずる。

【2173】Real  
ある実数は、キャラクタテレクリプト中のトークン “Real” のシンタクティックルに従いstringを生ずる。

【2174】Telenumbr  
テレナンバは、プラスサイン (“+”), テレナンバの国の属性、スペース (“ ”)、及びテレナンバの属性 “テレホン (telephone)” の連鎖を生ずる。

【2175】Time  
時間は、時間をひとまずカレンダー時間に変換し次にカレンダー時間をstringに変換する結果を生ずる。

【2176】4. 77 テレアドレス (Teleaddress)  
オブジェクト (参照される)

・ Teleaddress

Class

Teleaddress:

インタフェース = (...);

Constructin

initialize:

プロテクトされる op (provider: OctetStrin!|Nil;  
location: String!|Nil);

レスポンドの属性 “ルーティングアドバイス (routingAdvice)” をクリアし、その他の本来の属性を同じ名前のアークメント (アークメント “ロケーション (location)” 及び “プロバイダ (provider)”) にする。しかし後者のアークメントが0であれば、属性 “プロバイダ (provider)” は、現在のプレイスの割り当てられたテレアドレスの属性とされる。

【2177】パブリックインスタンスオペレーション  
location:  
String!|Nil;

レスポンドが割り当てられているかまたはそのようなコンストレイントが課された場合、レスポンドによって表されるプレイスのロケーションを表すstring、さもなければ0。

【2178】provider:  
OctetString!;

オクテットstringは、テレネームの属性 “オーソリティ (authority)” がそうするように、レスポンドが表すプレイスを含む領域のプロバイダを表す。

【2179】routingAdvice:  
List [OctetString!];

オクテットstringは、属性 “プロバイダ (provider)” がそうするように、そして優先順位の減少する順序に、移動領域のプロバイダを表す。

【2180】アダプテーション

isEqual

2つのテレアドレスは、それぞれの固有の属性に従って同一である。

【2181】4. 78 テレネーム (Telename)

オブジェクト (参照される)

・ Telename

Class

Telename

インタフェース = (…);

Construction

initialize

プロテクトされない op (authority, identity:

OctetString!|Nil);

レスポндаの固有の属性を同じ名前のアーギュメント (アーギュメント "オーソリティ (authority)" 及び "アイデンティティ (identity)" にする。しかし前者のアーギュメントは0であれば、属性 "オーソリティ (identity)" が現在のプロセスの割り当てられたテレネームの属性にされる。

【2182】Public Instance Attributes

authority:

OctetString!

レスポндаが表す名前付けされたオブジェクトのオーソリティを表すオクテットストリング。

【2183】identity:

OctetString!|Nil;

レスポндаが割り当てられているかそのようなコンストレントが課された場合には、レスポндаによって表される名前付けされたオブジェクトのアイデンティティを表すオクテットストリングであり、さもなければ0である。

【2184】アダプテーション

isEqual

2つのテレネームは、それぞれの固有の属性に従って同一である。

【2185】4. 79 テレナンバ (Telenumber)

オブジェクト (参照される)

・ Telenumber

Class

Telenumber

インタフェース = (…);

Construction

initialize

プロテクトされない op (countryAndTelephone: String!;

extension: String!|Nil);

レスポндаの属性 "エクステンション (extension)" を、同じ名前のアーギュメント (アーギュメント "エクステンション (extension)" ) とし、レスポндаの他の固有の属性を、アーギュメント "カントリアンドテレ

ホン (countryAndTelephone)" に変換可能なテレナンバの属性にする。

【2186】Public Instance Attributes

country

String!;

レスポндаに表すインストルメントが含まれている国または他の地域的な領域にC C I T Tが割り当てるコード。ストリングの各々のキャラクタは数字 ("0" - "9") である。

【2187】extension:

String!|Nil;

レスポндаの属性 "カントリ (country)" 及び "テレホン (telephone)" がそれ自身そのようにしない場合、レスポндаが表すインストルメントを明瞭に特定するテレホンの内線であり、さもなければ0である。ストリングの各々のキャラクタは、数字 ("0" - "9")、ハイフン (" - ") またはスペース (" ") である。

【2188】telephone:

String!;

属性 "カントリ (country)" が表す国または他の地理的な領域によってレスポндаが表すインストルメントに割り当てられた番号。ストリングの各々のキャラクタは、数字 ("0" - "9")、ハイフン (" - ") またはスペース (" ") である。

【2189】アダプテーション

isEqual

2つのテレネームは、全てのハイフン (" - ") 及びスペース (" ") が最初にそれらが除かれたかのように同一であるそれらの固有の属性に従って同一である。

【2190】Conversions

String

あるテレナンバを変換することによって生じうるストリングは、その属性 "エクステンション" が0であり、テレナンバを生ずる。

【2191】4. 80 チケット

オブジェクト (参照される)

・ チケットスタブ

・ Ticket

Class

Ticket:

インタフェース (TicketStub) = (…);

Construction

initialize:

プロテクトされない op (destinationName: Telename|Nil;

destinationAddrss: Teleaddress|Nil;

destinationClass: Citation|Nil;

maximumWait: Integer|Nil;

way: Way|Nil;

travelNotes: Object|Nil);

レスポндаの属性"desiredWait"及び"destinationPermit"を0にし、レスポндаの他の属性を同じ名前のアーギュメント(アーギュメント"destinationAddress"、"destinationClass"、"destinationName"、"maximumWait"、"travelNotes"及び"way")にする。

【2192】Public Instance Attributes

desiredWait:

Integer|Nil;

そのようなコンストレイントが課せられた場合、レスポндаが規定するトリップが完了されるべき時間とトリップがリクエストされる時間との間の負数でない秒数の最大の所望される差であり、その他の場合は0である。

【2193】destinationAddress

Teleaddress|Nil;

そのようなコンストレイントが課せられた場合、レスポндаが定義するトリップの目的点のテレアドレスであり、さもなければ0である。

【2194】destinationClass:

Citation|Nil;

そのようなコンストレイントが課せられれば、レスポндаが定義するトリップの目的点が1つのメンバであるクラス"プレイス(place)"のサブクラスへのサイティションであり、さもなければ0である。

【2195】destinationName:

Telenam|Nil;

そのようなコンストレイントが課せられれば、レスポндаが定義するトリップの目的点のテレネームであるかまたは、さもなければ0である。

【2196】destinatonPermit:

Permit|Nil;

ローカルパーミットがエージェントの現在の一時的なパーミットと相違していれば、レスポндаを使用するエージェントはエージェントの目的点において持つべきローカルパーミットであり、エージェントが持たなければ、エージェントの固有のパーミットであり、さもなければ0である。

【2197】maximumWait:

integer|Nil;

もしそのようなコンストレイントが課せられれば、レスポндаが定義するトリップが終了すべき時間とトリップが要求する時間との間の秒で表した最大のパーミットされる差であり、またはさもなければ0である。整数はもし供給されれば負数ではないものとする。

【2198】アダプテーション

isEqual

2つのチケットはチケットの固有の属性及びチケットのスタブに従って同一である。

【2199】4. 8 1 チケットスタブ(Ticket Stu

b)

オブジェクト(参照される)

・ Ticket Stub

Class

TicketStub

インタフェース = (...);

Construction

initialize:

プロテクトされない op (way: Way|Nil; travelNotes: Object|Nil);

レスポндаの固有の属性を、同じ名前のアーギュメント(アーギュメント"トラベルノート(travelNotes)"及び"ウェイ(way)")とする。

【2200】Public Instance Attributes

travelNotes:

Object|Nil;

レスポндаを保持するエージェントの専用的な使用のためのオブジェクト。

【2201】way

Way|Nil;

もしそのようなコンストレイントが課せられれば、トリップの目的点に向かって取るべき道でありさもなければ0である。チケットスタブにおいてトリップの原点への戻り道。

【2202】アダプテーション

isEqual

2つのチケットスタブはそれぞれの固有の属性に従って同一である。

【2203】4. 8 2 タイム(Time)

オブジェクト(参照される)

・ Time(順序付けされ、変更されない)

Class

Time:

インタフェース (Object, Ordered, Unchanged) = (...);

Constlruction

initialize

レスポндаを現在の時間すなわち現在のプレイスにおいての時間とする。

【2204】パブリックインスタンスオペレーション

adjust:

op (seconds: Integer) Time;

レスポндаの後の要求された秒数(アーギュメント"セコンズ(seconds)")であり、同一の恒久的オフセット及び季節的オフセットを特定する時間はリターンする。

【2205】注: 整数が負数であればリターンされる時間はレスポндаに先行する。

【2206】interval:

op (subtrahend: Time) Integer;

被減数であるレスポンスと減数である時間（アーギュメント”サブトラヘンド（subtrahend）”）との間の秒で表した算術差をリターンする。

【2207】注：結果は2つの絶対時間との間の距離を表す。

#### 【2208】Adaptations

isAfter

ある時間が別の時間の次になるのは、第2のものが特定する絶対時間点に第1の特定する絶対時間点が後続する場合にのみ生ずる。

#### 【2209】isBefore

ある時間が別の時間の前になるのは、第2のものが特定する絶対時間点よりも第1のものが特定する絶対時間点が先行する場合にのみ生ずる。

#### 【2210】isEqual

2つの時間はどちらも他のものの前でない場合にのみ同一である。

#### 【2211】Conversions

CalendarTime

カレンダー時間は、同一の時点を表す時間と同一の恒久的及び季節的オフセットとを生ずる。

#### 【2212】4.83 トリップエクセプション (Trip Exception)

オブジェクト（参照される）

・例外（変更されない）

・”Trip Exception”

クラス

TripException:

アブストラクト・インタフェース(例外)=(...);

Construction

initialize:

アブストラクト・オペレーション(ticketStub: TicketStub);

レスポンスの固有の属性を同じ名前のアーギュメント（アーギュメント”ticketStub”）とする。

#### 【2213】Subclassess

DestinationUnavailable:

インタフェース(トリップの例外)=0;

ネットワークが区画されているためトリップの目的点は到達不可能である。従って、目的点は将来到達されうるかもしれない。

#### 【2214】DestinationUnknown:

インタフェース(トリップの例外)=0;

トリップの目的点は特定できないので到達不可能である。

#### 【2215】OccupancyDenied:

インタフェース(トリップの例外)=0;

トリップの目的点はエージェントの占有従って到達を拒絶する。

#### 【2216】TicketExpired:

インタフェース(トリップの例外)=0;

トリップに対するチケットの属性”maximumWait”によって要求された秒数内にはトリップの目的点には到達できない。

#### 【2217】WayUnavailable:

インタフェース(トリップの例外)=0;

トリップの出発点はトリップについて要求された航路が欠如している。

#### 【2218】Public Instance Attributes

ticketStub:

sealed TicketStub;

成功しなかったトリップから結果したチケット・スタブ。

#### 【2219】Adaptations

isEqual

2つのトリップの例外はそれぞれの固有の属性に従って同一である。

#### 【2220】4.84 アンチェンジド (Unchanged)

クラス

Unchanged:

アブストラクト・インタフェース0=0;

Adaptations

copy

変更されないオブジェクトのコピーはオリジナルである。

#### 【2221】4.85 アネクスペクティドエクセプション (Unexpected Exception)

オブジェクト（参照される）

・例外（変更されない）

・プログラミングの例外

・カーネルの例外

・実行の例外

・”Unexpected Exception”

クラス

UnexpectedException:

インタフェース(実行の例外)=(...);

Construction

initialize

アブストラクト・オペレーション(exception: Exception);

レスポンスの固有の属性を同じ名前のアーギュメントとする。（アーギュメント”Exception”）

Public Instance Attributes

exception:

リードオンリの例外。

【2222】あるフィーチャが宣言しなかったが投出した、レスポンスの原因である例外。

#### 【2223】4.86 アンムーブド (Unmoved)

クラス

Unmoved

アブストラクト・インタフェース 0 = 0 ;

4. 87 ベリファイド (Verified)

クラス

Verified

アブストラクト・インタフェース 0 = (···) ;

Public Instance Operations

verify:

アブストラクト op () プリアン;

レスポンドがコンシステントであるか否かの表示をする。

【2224】4. 88 ウェイ (Way)

オブジェクト (参照される)

・Way

クラス

Way:

インタフェース = (···) ;

Construction

initialize:

プロテクトされない op (ネーム: Telename|Nil; 手段:

Means|Nil; オセンティケータ: Authenticator|Nil);

レスポンドの固有の属性を同じ名前のアーギュメント (アーギュメント "authenticator"、"means" 及び "name") とする。

【2225】Public Instance Attributes

authenticator:

オセンティケータ|Nil;

そのようなコンストレイントが課せられるならば、航路が通過する領域に入るために用いられるためのオセンティケータであり、さもなければ 0 である。

【2226】means:

手段|Nil;

もしそのようなコンストレイントが課せられれば、航路が通過する領域をアクセスすべき手段であり、さもな

ければ 0 である。

【2227】name:

テレネーム|Nil;

もしそのようなコンストレイントが課せられれば、航路が通過する領域のテレネームであり、さもなければ 0 である。

【2228】Adaptations

isEqual

2つの航路はそれぞれの固有の属性に従って同一である。

【2229】5 テレスクリプトシンタックス (TELESCRIPT SYNTAX)

"telescript" はプロシージャの符号 (エンコーディング) である。一般にテレスクリプトの抽象的なシンタックスと特にキャラクタ及び2進テレスクリプトの具体的なシンタックスとはこのアペンディックスのこのセクションにおいて定義される。

【2230】注: キャラクタ・テレスクリプト、人々によって、2進テレスクリプトは機械によって、例えばオブジェクトをそれらの間に搬送する際に用いられる。2進テレスクリプトは機能的の同等のキャラクタ・テレスクリプトよりも常にオクテットで表した長さが著しくない。

【2231】5. 1 テレスクリプト (Telescript)

テレスクリプトは、第1にプリフェースをそして第2テレスクリプトが符号化するプロシージャを含む一連のトークンである。プリフェースは、テレスクリプトがそれに対して適合される命令セットのメジャーバージョン及びマイナーバージョンを特定する。

【2232】テレスクリプトは、以下に示すシンタックティック・ルール及びそれに付随したセマンティック・ルールに従う。BNFで表した規則は、ブラケットによって任意のトークンを囲む (" [" and " ] ")

```

Telescript ::= Preface Procedure
    Body ::= [ExecutedObject Body] ExecutedObject ::= Bit|BitString|Boolean|
    Character|Identifier|Integer|Octet|OctetString|Mark|
    Modifier| Nil|Procedure|Mark|
    Real|Selector|String
    Bit ::= BitZero| BitOne
    Boolean ::= Booleanalse|BooleanTrue
    Modifier ::= ModifierDemarcate|
    ModifierGetClass|
    ModifierGetProperty|
    ModifierGetVariable|
    ModifierMention|
    ModifierSetAttribute|
    ModifierSetProperty|
    ModifierSetVariable|
    ModifierUseStack|
    Selector ::= SelectorBreak|
    SelectorClient|

```

SlectorContinue|  
 SlectorEscalate|  
 SlectorPlace|  
 SlectorProcess|  
 SlectorSelf|  
 SlectorSucceed

テレスクリプトはそのための規定がシンタックティック・ルールに含まれると考えられるコメントを含めうる。”comment”はテレスクリプトが符号化するプロシージャに影響しないストリングである。1以上のコメントは、スクリプトの第1トークンの前、その最後のトークンの後そして任意の2つの隣接したトークンの間に出現しうる。

【2233】注：あるコメントは、通常は、ヒトのテレスクリプトの1つの様相を通常記述する。

【2234】注：このアベンディックスに規定された命令セットのメジャーバージョン及びマイナーバージョンは、それぞれ0(“0”)及び8(“8”)と特定される。

【2235】5.2 キャラクタテレスクリプト(Character Telescript)  
 “character telescript”はストリングとして符号化されたプロシージャである。一般にテレスクリプトの抽象的なシンタックティック・ルールは、特にこのセクションにおいてキャラクタ・テレスクリプトにおいて具体化されている。

【2236】あるキャラクタ・テレスクリプト中の各々のトークンは1以上の文字である。キャラクタ・テレスクリプトは、トークン従ってキャラクタを連鎖化することによってそしてトークンの間にブレイク(break)キャラクタを挿入することによって得られる。

【2237】ブレイク・キャラクタはスペース(“ ”)、水平タブまたはラインフィードである。1以上のそのようなキャラクタは、最初のトークンの前、最後のトークンの後または2つの任意の隣接したトークンの間に表れうる。最後の場合には2つの隣接したトークンがどちらも“BinDigit”または“ヘックスディジットペア”である場合を除いて、これら2つのトークンが数字(“0”-“9”)及び文字からなる場合には少なくとも1つのキャラクタは表れなければならない。

【2238】数字サイン(“#”)及びキーワード“telescript”でもって始まるターミナルは、ケース(大文字-小文字)を感知しない。アッパーケース(大文字)キャラクタがこのアベンディックスに表れるが、ローア-エース(小文字)キャラクタまたは2つのものの混合がプログラミングにおいて使用されうる。

【2239】5.2.1 プリフェイスとコメント(Preface and Comment)  
 Preface  
 プリフェースはこれらの規則に従う。

【2240】プリフェイス:=telescript Version  
 バージョン :=numerals :numerals

第1の“numerals”は、キャラクタ・テレスクリプトがそれに従うところの命令セットのメジャーバージョンを特定し、第2の“numerals”はマイナーバージョンを特定する。

【2241】Comment

コメントはこの規則に従う。

【2242】Comment::=/\*comment1\*///comment2

注：コメントのためのシンタックス・ルールは、C++プログラミング言語のものである。

【2243】5.2.2 エグゼキュートオブジェクト  
 (Executed Objects)

Bit

ビットは、ビットの値を個別に符号化する次の規則に従う。

【2244】BitZero::=#b'0'

BitOne::=#b'1'

BitString

ビット・ストリングは次の規則に従う。

【2245】BitString::=#b"BinDigits"

BinDigits:=[BinDigit BinDigits]

i番目の“BinDigit”はビット・ストリングの位置“i”においてのビットを表す。

【2246】Boolean

ブーリアンは、ブーリアンの値を個別に評価するこれらの規則に従う。

【2247】BooleanFalse::=#false

BooleanTrue::=#true

Character

キャラクタは次の規則に従う。

【2248】Character::=#C'characters'

“characters”は1つの文字を表すものとする。各々のキャラクタはそれ自身を表すが、クラス“String”に規定した例外がある。

【2249】Identifier

特定子はこの規則に従う。

【2250】Identifier::=identifier|]

“Identifier”は、識別子のテキストである。右の角括弧[“”]は、オペレーション“new”を表す。

【2251】Integer

整数は次の規則に従う。

【2252】Integer::=[-]numerals

Mark



マークは次の規則に従う。

【2253】Mark ::= #mark|

注：左角括弧 [ " [ " ] ] は、右角括弧 [ " ] " ] とともに、コンベンションによって、可能な使用のために設けられる（上記のトークン " Identifier" 参照）

Modifier

モディファイヤはモディファイヤの値を個別に符号化する次の規則に従う。

【2254】ModifierDemarcate ::= @

ModifierGetClass ::= :

ModifierGetProperty ::= %

ModifierGetVariable ::= \$

ModifierMention ::= '

ModifierSetAttribute ::= =

ModifierSetProperty ::= =%

ModifierSetVariable ::= =\$

ModifierUseStack ::= -

Nil

0 は次の規則に従う。

【2255】Nil ::= #nil

Octet

オクテットは次の規則に従う。

【2256】Octet ::= #0' hexDigitPair'

" hexDigitPair" のMSBは、オクテットのビット7を表し、次が0を表す。

【2257】OctetString

オクテット・ストリングは次の規則に従う。

【2258】OctetString ::= #0"HexDigitPairs"

HexDigitPairs ::= [hexDigitPair HexDigitPairs]

第1の " hexDigitPair" は、オクテット・ストリングの位置1におけるオクテットを表し、第2のものは位置2におけるオクテットを表し、以下同様である。

【2259】各々の " hexDigitPair" のMSBは、このオクテットのビット7を表し、LSBはビット0を表す。

【2260】Procedure

オクテットは次の規則に従う。

【2261】Procedure ::= [Body]

QualifiedIdentifier

有資格識別子は次の規則に従う。

【2262】QIdentifier ::= identifier::identifier

第1のアイデンティファイヤはテキストを表し、

Real

実数は次の規則に従う。

【2263】Real ::= Number [Exponent]

Number ::= Integer·numerals|

Integer·|

·numerals

Exponent ::= E Integer

実数の整数部分は、もし存在していれば、" Number" の " Integer" であるか、さもなければ0である。実数の分数部分は、もし存在していれば " numerals" であるか、さもなければ0である。

【2264】" Exponent" 中の " Integer" は、もし存在していれば10のべき乗である。

【2265】Selector

セクタは、セクタの値を個別に符号化する次の規則に従う。

【2266】SelectorBreak ::= #break

SelectorClient ::= #client

SelectorContinue ::= #continue

SelectorEscalate ::= #escalate|'

SelectorPlace ::= #here

SelectorProcess ::= #process

SelectorSelf ::= #self|\*

SelectorSucceed ::= #succeed

ストリングは次の規則に従う。

【2267】String ::= [#C]"characters"

" characters" は0またはそれ以上のキャラクタを表す。各々のキャラクタはそれ自身を表すが、例外として逆スラッシュ (" \") 及び4つの16進デジットを (" 0" - " 9", " A" - " F") が、そのユニコード・コードがデジットの表す符号のない整数であるキャラクタ、指示されたキャラクタについて表A. 6に示したキャラクタ対を表す。

【2268】

【表6】

表 A. 6	
キャラクタシーケンス	単一キャラクタ
\b	バックスペース (Backspace)
\f	紙送り (Form feed)
\n	行送り (newline)
\r	改行 (Carriage return)
\t	水平タブ (Horizontal tab)
\v	垂直タブ (Vertical tab)
\"	クォーテーションマーク (" " )
\\	逆スラッシュ (" \ " )

## characters

0またはそれ以上のキャラクタは各々プリミティブ・キャラクタまたはスペース（" "）を表す。引用符（"）は逆スラッシュ（" \ "）によって先行される。逆スラッシュは、上記のトークン"String"の下に見いだされる表中のキャラクタ・シーケンスの一部分であるかまたは4つの16進デジットに先行するものとする。

【2269】comment1

スラッシュ（" \*/"）が直後に続いていないアスタリスクを含まない0またはそれ以上のキャラクタ。

【2270】注："comment1"は特別な意味なしに"/ \* "または"/ / "を包括することができるので、"comment2"を包括しうる。

【2271】comment2

ライン・フィードを含まない0またはそれ以上のキャラクタ。

【2272】注："comment2"は特別な意味なしに"/ \* "、" \*/ "または"/ / "を包括しうるので、"comment1"を包括しうる。"comment2"は、同じ行において他のトークンによって後続されることはできないが、それは"comment2"がこれらを含むものと考えられるからである。

【2273】hexDigitPair

16進デジットの対（"0" - "9", "A" - "F"）

## identifier

識別子のテキストと同様に拘束（コンストレイン）されたキャラクタ。

【2274】numerals

1以上の数字（"0" - "9"）

## Binary Telescript

"binary telescript"はオクテット・ストリングとして符号化されたプロシージャである。一般にテレスク립トの抽象的なシンテック・ルールは、特にこのセクションにおいて2進テレスク립トについて具体化されている。

【2275】2進テレスク립ト中の各々のトークンは1以上のオクットである。2進テレスク립トは、トークン従ってオクテットを連鎖させることによって得られる。

【2276】5. 3. 1 プリフェイスとコメント（Preface and Comment）

## Preface

プリフェースは次の規則に従う。

【2277】Preface ::= Version

Version ::= unsignedNumber unsignedNumber

第1及び第2の"unsignedNumber"は、キャラクタ・テレスク립トがそれに従う命令セットのマージャーバージョン及びマイナーバージョンをそれぞれ特定する。

【2278】Comment

コメントは次の規則に従う。

【2279】Comment ::= tab unsignedNumber Characters

"unsignedNumber"は文字として入力されるべきオクテットの数である。

【2280】5. 3. 2 エグゼキューティドオブジェクト（Executed Objects）

## Bit

ビットはビットの値を個別に符号化する次の規則に従う。

【2281】BitZero ::= tag

BitOne ::= tag

## BitString

ビット・ストリングは次の規則に従う。

【2282】BitString ::= tag unsignedNumber unsignedNumber Octets

Octets ::= [octet Octets]

第1の"unsignedNumber"は、最後のオクテットのビット・ストリングの各ビットの、範囲[1, 8]中の数である。このオクテット中の他のビットは意味を持たないので定義されない。第2の"unsignedNumber"は、トークン・オクテット中のトークン"Octets"の発生回数"n"である。オクテットが[1, n]において番号付けされていれば、ビット・ストリング中のその位置が"i"であるビットは、番号(1 + (i - 1) / 8)の番号を有するオクテットのビット(7 - ((i - 1) mod 8))である。

【2283】Boolean

ブーリアンはブーリンの値を個別に符号化するこれらの規則に従う。

【2284】BooleanFalse ::= tag

BooleanTrue ::= tag

## Character

キャラクタはこれらの規則に従う。一般的及び特別の符号化が規定されている。後者は8ビットのキャラクタについて用いる。

【2285】Character ::= GeneralCharacter |

SpecialCharacter

GeneralCharacter ::= tag octet octet

SpecialCharacter ::= tag octet

1または2のオクテットは、キャラクタのユニコード・コードであるキャラクタを符号化する。特別な符号化において整数のMSB及びLSBがオクテットのMSB及びLSBである。一般的な符号化において整数のMSB及びLSBは、それぞれ第1のオクテットのMSBと第2のオクテットのLSBである。

【2286】Identifier

識別子は次の規則に従う。1つの一般的な符号化及び2つの特殊な符号化が規定されている。後者の1つは識別

子のテキストが予め規定されたクラスまたはフィーチャの識別子のテキストに等しい場合にのみ使用する。

【2287】 Identifier ::= GeneralID |

PredefinedClassID |

PredefinedFeatureID

GeneralID ::= tag unsignedNumber characters

PredefinedClassID ::= tag unsignedNumber

PredefinedFeatureID ::= tag unsignedNumber

一般的な符号化において "unsignedNumber" 及び "characters" は直接にテキストを表す。前者は、後者として認識されるべきビットのオクテットの数である。特別の符号化において、"unsignedNumber"、セクションの5.4の表の1つから引き出されるコードによって、識別子のテキストを関節に表している。

【2288】 Integer 整数は次の規則に従う。1つの一般的な符号化といくつかの特殊な符号化が規定されている。後者は、-1、0 及び +1 で用いる。

【2289】

Integer ::= GeneralInteger | SpecialInteger

GeneralInteger ::= tag signedNumber

SpecialInteger ::= IntegerMinusOne |

IntegerZero |

IntegerPlusOne

IntegerMinusOne ::= tag

IntegerZero ::= tag

IntegerPlusOne ::= tag

サインドナンバは整数である。

【2290】 Mark

マークは次の規則に従う。

【2291】 Mark ::= tag

Modifier

モディファイヤは、モディファイヤの値を個別に符号化する次の規則に従う。

【2292】 ModifierDemarcate ::= tag

ModifierGetClass ::= tag

ModifierGetProperty ::= tag

ModifierGerVariable ::= tag

ModifierMention ::= tag

ModifierSetAttribute ::= tag

ModifierSetProperty ::= tag

ModifierGerVariable ::= tag

Nil

0 は次の規則に従う。

【2293】 Nil ::= tag

Octet

オクテットは次の規則に従う。

【2294】 Octet ::= tag octet

"octet" の MSB はオクテットのビット7、LSB ビット0、を表す。OctetString オクテット・ストリングは次の規則に従う。

【2295】

OctetString ::= tag unsignedNumber Octets

Octet ::= [octet Octets]

"アンサインドナンバ" は、"オクテット" 中の "オクテット" の発生回数である。"Octets" 中の第1 "Octet" は、オクテット・ストリングの位置1 においてのオクテットを表し、第2の "Octet" は、位置2 においてのオクテットを表し以下同様である。各々の "Octet" の MSB は、そのオクテットのビット7を表し、LSB はビット0を表す。

【2296】 Procedure

プロシージャは次の規則に従う。

【2297】 Procedure ::= tag unsignedNumber Body

"unsignedNumber" は、各々プロシージャのアイテムを表す "Body" 中の "エグジュキューティッドオブジェクト" の発生回数を符号化する。アイテムは、プロシージャにおいて位置の増大する順序において示されている。

【2298】 QualifiedIdentifier

有資格識別子は次の規則に従う。

【2299】

QIdentifier ::= tag Identifier Identifier

第1の "Identifier" は、テキストを、第2のものはクオリファイヤをそれぞれ表す。

【2300】 Real

実数は、次の規則に従う。

【2301】 Real ::= tag signedNumber signedNumber

第1の "signedNumber" は、仮数 "m" を表し、第2のものは指数 "e" を表す。実数は、 $m \times 10^e$  である。

【2302】 Selector

セレクトはセレクトの値を個別に符号化する次の規則に従う。

【2303】 SelectorBreak ::= tag

Selector Client ::= tag

Selector Continue ::= tag

Selector Escalate ::= tag

Selector Place ::= tag

Selector Process ::= tag

Selector Self ::= tag

Selector Succeed ::= tag

String

ストリングは次の規則に従う。

【2304】

String ::= tag unsignedNumber characters

"unsignedNumber" は、キャラクタとして認識されるべきオクテットの数である。オクテットのシンタックス及びセマンティックスは、ISO/IEC10646 [10646] コンパクション・メソッド5に従うが、例外として、これらによって表されたキャラクタは、ユニコードのキャラクタに制限される。

【2305】注：ストリングがASCIIキャラクタのみを含む場合、“characters”はそのASCII表示を1オクテットについて1キャラクタで表し、各々のオクテットのビット7は0である。

【2306】5. 3. 3 他の非ターミナル (Other Non-terminals)

characters

0またはそれ以上のオクテット

number

1から5までについて、“N”は整数“I”を符号化するオクテットである。“I”の符号化“E”は数の使用が、“I”が負数でないことを主張するならば、定義されてなくその他の場合には2の補数である。与えられた“I”、“N”は可及的に小さくする。

【2307】“E”は次のようにして定められる。

【2308】・ 最初のオクテットの値が[0, BF<sub>16</sub>]の範囲であれば、“N”は1であり、“E”のMSB及びLSBは、その単一のオクテットのそれぞれビット7及びビット0である。

【2309】・ 第1のオクテットがEO<sub>16</sub>, E2<sub>16</sub>, E4<sub>16</sub>またはE6<sub>16</sub>であれば、“N”は、2、3、4または5であり、“E”のMSB及びLSBは、第1オクテットが00<sub>16</sub>とされた場合に最初のオクテットのビット7及び最後のオクテットのビット0となる。

【2310】・ 最初のオクテットがE1<sub>16</sub>, E3<sub>16</sub>, E5<sub>16</sub>またはE7<sub>16</sub>であった場合、Nは、2、3、4または5であり、“E”のMSB及びLSBは、第1オクテットがFF<sub>16</sub>とされた場合に最初のオクテットがビット7及び最後のオクテットのビット0となろう。

【2311】注：[C0<sub>16</sub>, DF<sub>16</sub>]及び[E8<sub>16</sub>, FF<sub>16</sub>]の各範囲内の第1オクテットの値は留保されている。

【2312】octet

1オクテット。

【2313】signedNumber

“I”が正または負であることができ、従ってが符号を

持ち(上記参照)場合の“number”。

【2314】注：“N”が1であれば、“I”は[-64, 127]の範囲内にある。

【2315】“N”が2であれば、“I”は[-128, 127]の範囲内にある。

【2316】“N”が3であれば、“I”は[-32768, 32767]の範囲内にある。

【2317】“N”が4であれば、“I”は[-8388608, 8388607]の範囲内にある。

【2318】“N”が5であれば、“I”は[-2147483648, 2147483647]の範囲内にある。

【2319】tag

タグの意味は次のセクションに与えられる場合の“signedNumber”。

【2320】unsignedNumber

“I”は負数でなく、従って“E”が符号を持たない場合(上記参照)の“number”。

【2321】注：“N”が1であれば、“I”は[0, 19]の範囲内にある。

【2322】“N”が2であれば、“I”は[0, 255]の範囲内にある。

【2323】“N”が3であれば、“I”は[0, 65535]の範囲内にある。

【2324】“N”が4であれば、“I”は[0, 16777215]の範囲内にある。

【2325】“N”が5であれば、“I”は[0, 4294967295]の範囲内にある。

【2326】5. 4 数値コード

2進テレスクリプトは次の数字コードで使用する。

【2327】5. 4. 1 プレデファインディッドクラス

表A. 7は、クラス“Executed”の予め規定されていないサブクラスへの識別コードを割り当てる。

【2328】

【表7】

表 A. 7	
1	ビット (Bit)
2	ビットストリング (BitString)
3	ブーリアン (Boolean)
4	キャラクタ (Character)
5	アイデンティファイ (Identifier)
6	整数 (Integer)
7	マーク (Mark)
8	モディファイヤ (Modifier)
9	ニル (Nil)
1 0	8 進 (Octet)
1 1	8 進数ストリング (OctetString)
1 2	プロシージャ (Procedure)
1 3	Q アイデンティファイヤ (QIdentifier)
1 4	実数 (Real)
1 5	セレクト (Selector)
1 6	ストリング (String)

表 A. 8 は、他の予め規定されていないメジャークラ      【 2 3 2 9 】  
 スへの識別コードを割り当てる。      【表 8】

表 A. 8	
2 0	エージェント (Agent)
2 1	アソシエーション (Association)
2 2	アトリビュート (Attribute)
2 3	オーセンティケータ (Authenticator)
2 4	カレンダータイム (CalendarTime)
2 5	ケースド (Cased)
2 6	サイテーション (Citation)
2 7	サイティド (Cited)
2 8	クラス (Class)
2 9	クラス定義 (ClassDefinition)
3 0	クラス例外 (ClassException)
3 1	コレクション (Collection)
3 2	コレクション例外 (CollectionException)
3 3	コンストレイン (Constrain)
3 4	コンストレイント'ディクショナリ (ConstrainedDictionary)
3 5	コンストレイント'リスト (ConstrainedList)
3 6	コンストレイント'セット (ConstrainedSet)
3 7	コンストレイント (Constraint)
3 8	コンタクト (Contact)
3 9	コンタクティド (Contacted)
4 0	ディクショナリ (Dictionary)
4 1	エクセプション (Exception)
4 2	エグゼキューティド (Executed)
4 3	エグゼキューション (ExcutionException)
4 4	-
4 5	フィーチャ (Feature)

【 2 3 3 0 】

【表 9】

表 A. 8	
4 6	ハッシュド (Hashed)
4 7	インプ レメンテーション (Implementation)
4 8	インターチェンジド (Interchanged)
4 9	インターフェース (Interface)
5 0	カーネルエクセプション (KernelException)
5 1	レキシコン (Lexicon)
5 2	リスト (List)
5 3	ミーンズ (Means)
5 4	ミーティング エクセプション (MeetingException)
5 5	ミーティング プレイス (MeetingPlace)
5 6	メソッド (Method)
5 7	ミセルニアスエクセプション (MiscellaneousException)
5 8	ネームド (Named)
5 9	ナンバー (Number)
6 0	オブジェクト (Object)
6 1	オペレーション (Operation)
6 2	オーダード (Ordered)
6 3	パッケージ (Package)
6 4	パターン (Pattern)
6 5	パーミット (Permit)
6 6	ペティション (Petition)
6 7	ペティションド (Petitioned)
6 8	プレイス (Place)
6 9	プリミティブ (Primitive)

【2331】

【表10】

表 A. 8	
7 0	プリミティブエクセプション (PrimitiveException)
7 1	プロセス (Process)
7 2	プロセス例外 (ProcessException)
7 3	プロセス例外 (ProcessException)
7 4	ランダムストリーム (RandomStream)
7 5	-
7 6	リファレンス (Referenced)
7 7	-
7 8	-
7 9	リソース (Resouce)
8 0	セット (Set)
8 1	スタック (Stack)
8 2	ストリーム (Stream)
8 3	テレアドレス (Teleaddress)
8 4	テレネーム (Telename)
8 5	テレナンバ (Telenumber)
8 6	チケット (Ticket)
8 7	チケットスタブ (TicketStub)
8 8	タイム (Time)
8 9	トリップ例外 (TripException)
9 0	アンチェンジド (Unchanged)
9 1	アンエクスペクティドエクセプション (UnexpectedException)
9 2	アンムーブ (Unmove)
9 3	ベリファイド (Verified)
9 4	ウェイ (Way)

表 A. 9 は、予め規定されていないマイナークラスへの  
識別コードを割り当てる。

【 2 3 3 2 】

【 表 1 1 】

表 A. 9	
1 0 0	プリミティブインアデクエイト (PrimitiveInadequate)
1 0 1	ア-キ-メントインバ-リッド (ArgumentInvalid)
1 0 2	ア-キ-メントミッシング (ArgumentMissing)
1 0 3	アトリビュートリード-オンリ (AttributeReadOnly)
1 0 4	クラスアブストラクト (ClassAbstract)
1 0 5	クラスアンアベイラブル (ClassUnavailable)
1 0 6	クラスアンデファイン (ClassUndefined)
1 0 7	クラスシールド (ClassSealed)
1 0 8	コンディションアンアベイラブル (ConditionUnavailable)
1 0 9	コンディションアンデファイン (ConditionUndefined)
1 1 0	コンバージョンアンアベイラブル (ConversionUnavailable)
1 1 1	コピーアンアベイラブル (CopyUnavailable)
1 1 2	デスティネーションアンアベイラブル (DestinationUnavailable)
1 1 3	デスティネーションアンノーン (DestinationUnknown)
1 1 4	ディビジョンバイゼロ (DivisionByZero)
1 1 5	-
1 1 6	エスカレーションインバ-リッド (EscalationInvalid)
1 1 7	フィーチャリデファイン (FeatureRedefined)
1 1 8	フィーチャアンアベイラブル (FeatureUnavailable)
1 1 9	フィーチャシールド (FeatureSealed)
1 2 0	フィーチャアンデファイン (FeatureUndefined)
1 2 1	インターナルエクセプション (InternalException)
1 2 2	アイテム重複 (ItemDuplicated)
1 2 3	アイテムインバ-リッド (ItemInvalid)

【2333】

【表12】



表 A. 9	
1 2 4	キーデュプリケイティド (KeyDuplicated)
1 2 5	キーインバリッド (KeyInvalid)
1 2 6	ループミッシング (LoopMissing)
1 2 7	マークミッシング (MarkMissing)
1 2 8	ミーティングデニード (MeetingDenied)
1 2 9	ミーティングデュプリケイティド (MeetingDuplicated)
1 3 0	ミーティングインバリッド (MeetingInvalid)
1 3 1	ミクスインディスアロート (MixInDisallowed)
1 3 2	オブジェクトアンペアード (ObjectsUnpaired)
1 3 3	オブジェクトアンイニシャライズド (ObjectUninitialized)
1 3 4	オキュパニイデニード (OccupanyDenied)
1 3 5	パスセイジインバリッド (PassageInvalid)
1 3 6	パターンインバリッド (PatternInvalid)
1 3 7	パーミットエクスハースト (PermitExhausted)
1 3 8	パーミットバイオリイティド (PermitViolated)
1 3 9	ペティションエクスピアード (PetitionExpired)
1 4 0	ポジションインバリッド (PositionInvalid)
1 4 1	プロセスノットカレント (ProcessNotCurrent)
1 4 2	プロセスノットピア (ProcessNotPeer)
1 4 3	プロパティアンデフィニート (PropertyUndefined)
1 4 4	リファレンスプロテクティド (ReferenceProtected)
1 4 5	リファレンスボイド (ReferenceVoid)
1 4 6	レスポンスミッシング (ResponderMissing)
1 4 7	リザルトインバリッド (ResultInvalid)

【2 3 3 4】

【表 1 3】

表 A. 9	
1 4 8	リザルトミッシング (ResultMissing)
1 4 9	セレクトラデュプリケイティド (SelectorDuplicated)
1 5 0	スタックデプレット (StackDepleted)
1 5 1	ステートインプロパー (StateImproper)
1 5 2	スーパークラスインバリッド (SuperclassInvalid)
1 5 3	チケットエクスパイアド (TicketExpired)
1 5 4	バリエーブルアンデフィニート (VariableUndefined)
1 5 5	ウェイアンバイル (WayUnavailable)
1 7 9	リソースアンバイル (ResouceUnavailable)
1 8 0	シードインバリッド (SeedInvalid)

5. 4. 2 プレデファインディッドフィーチャーズ (Predefined Features)

表 A. 10 は、予め規定されたオペレーションへの識別

コードを割り当てる。

【2 3 3 5】

【表 1 4】

表 A. 10	
1	アブス (abs)
2	アッド (add)
3	アジャスト (adjust)
4	アンド (and)
5	キャッチ (catch)
6	シーリング (ceiling)
7	チェンジ (change)
8	クリア (clear)
9	コンバート (convert)
10	コピー (copy)
11	ディファレンス (difference)
12	ディスカード (discard)
13	ディバイド (divid)
14	ドウ (do)
15	ドロップ (drop)
16	アイザ (either)
17	エンタリング (entering)
18	エグザミン (examine)
19	エクスクルード (exclude)
20	エグジッティング (exiting)
21	ファイナライズ (finalized)
22	ファインド (find)
23	フロア (floor)
24	ゲット (get)
25	グローバルイズ (globalize)
26	ゴー (go)
27	イフ (if)
28	インクルード (include)
29	イニシャライズ (initialize)

【2336】

【表15】

表 A. 10	
3 0	インターセクション (intersection)
3 1	インターバル (interval)
3 2	イズアフタ (isAfter)
3 3	イズビフォア (isBefore)
3 4	イズイコール (isEqual)
3 5	イズインスタンス (isInstance)
3 6	イズメンバ (isMember)
3 7	イズセイム (isSame)
3 8	イズサブクラス (isSubclass)
3 9	ライブ (live)
4 0	ローカライズ (localize)
4 1	ループ (loop)
4 2	メイクラスイズ (makeClasses)
4 3	メイクロア (makeLower)
4 4	メイクアップ (makeUpper)
4 5	ミート (meet)
4 6	ミーティング (meeting)
4 7	モジュラス (modulus)
4 8	マルチプライ (multiply)
4 9	ネゲート (negate)
5 0	ニュー (new)
5 1	ノーマライズ (normalize)
5 2	ノット (not)
5 3	オア (or)
5 4	パート (part)
5 5	パートオール (partAll)

【2337】

【表16】

表 A. 1 0	
5 6	パーティング (parting)
5 7	ポップ (pop)
5 8	プロテクト (protect)
5 9	プッシュ (push)
6 0	プッシュアイテム (pushItem)
6 1	クォシエント (quotient)
6 2	レフ (ref)
6 3	リキー (rekey)
6 4	リピート (repeat)
6 5	リポジション (reposition)
6 6	リストリクト (restrict)
6 7	ロール (roll)
6 8	ラウンド (round)
6 9	セレクト (select)
7 0	センド (send)
7 1	セット (set)
7 2	ストリーム (stream)
7 3	サブスティテュート (substitute)
7 4	サブストリング (substring)
7 5	サブトラクト (subtract)
7 6	スワップ (swap)
7 7	ターミネイト (terminate)
7 8	スロー (throw)
7 9	トランスポーズ (transpose)
8 0	トランケート (truncate)

【2338】

【表17】

表 A. 1 0	
8 1	ユニオン (union)
8 2	ユース (use)
8 3	ベリファイ (verify)
8 4	ウエイト (wait)
8 5	ホワイル (while)
8 6	マックス (max)
8 7	ミニ (min)

表 A. 1 1 は、予め規定された属性への識別コードを割り当てる。

【2339】

【表18】

表 A. 1 1	
8 9	アドレス (address)
9 0	エージェントクラス (agentClass)
9 1	エージェントネーム (agentName)
9 2	アローワンス (allowance)
9 3	アージェント (argument)
9 4	オーセンティケイタ (authenticator)
9 5	オーサ (author)
9 6	オーソリティ (authority)
9 7	ブランド (brand)
9 8	キャンチャージ (canCharge)
9 9	キャンゴー (canGo)

【2340】

【表19】

表A. 11	
100	キャンブロクリート (canProcreate)
101	キャンリスタート (canRestart)
102	キャンセンド (canSend)
103	キャンターミネート (canTerminate)
104	サイテーション (citation)
105	クラス (Class)
106	クラスフィーチャ (ClassFeature)
107	クラスアイディー (ClassId)
108	クラスメソッド (classMethod)
109	コンディション (condition)
110	コンディショングル (conditions)
111	コンストレイント (constraint)
112	コンタクツ (contacts)
113	カントリー (country)
114	カレント (current)
115	デイ (day)
116	デイオブウィーク (dayOfWeek)
117	デイオブイヤー (dayOfYear)
118	デッドライン (deadline)
119	デザイヤドウエイト (desiredWait)
120	デ' スティネーションアドレス (destinationAddress)
121	デ' スティネーションクラス (destinationClass)
122	デ' スティネーションネーム (destinationName)
123	デ' スティネーションパーミット (destinationPermit)
124	ダイジェスト (digest)
125	ディエスティ (dst)

【2341】

【表20】

表 A. 1 1	
1 2 6	エクセプション (exception)
1 2 7	エクセプションズ (exceptions)
1 2 8	エクステンション (extension)
1 2 9	フロムメソッド (fromMethod)
1 3 0	ハッシュ (hash)
1 3 1	アワー (hour)
1 3 2	アイデンティファイ (identify)
1 3 3	インプレメンテーション (implementation)
1 3 4	インスタンスフィーチャーズ (instanceFeatures)
1 3 5	インスタンスメソッド (instanceMethods)
1 3 6	インターフェース (interface)
1 3 7	イズアブストラクト (isAbstract)
1 3 8	イズダーン (isDone)
1 3 9	イズローワ (isLower)
1 4 0	イズオプション (isOptional)
1 4 1	イズプロテクティド (isProtected)
1 4 2	イズパブリック (isPublic)
1 4 3	イズセット (isSet)
1 4 4	イズアッパ (isUpper)
1 4 5	キー (key)
1 4 6	レングス (length)
1 4 7	ロケーション (location)
1 4 8	メジャーエディション (majorEdition)
1 4 9	マキシマムウエイト (maximumWait)
1 5 0	ミーンズ (means)

【2 3 4 2】

【表 2 1】

表A. 1 1	
1 5 1	マイナーエディション (minorEdition)
1 5 2	ミニット (minute)
1 5 3	マンス (month)
1 5 4	ネーム (name)
1 5 5	ネクスト (next)
1 5 6	オブクラス (ofClass)
1 5 7	パッセージ (passage)
1 5 8	パーミット (permit)
1 5 9	プライオリティ (priority)
1 6 0	プライベートクラス (privateClasses)
1 6 1	プロシージャ (procedure)
1 6 2	プロパティズ (properties)
1 6 3	プロバイダ (provider)
1 6 4	パブリッククラシズ (publicClasses)
1 6 5	リザルト (result)
1 6 6	ルーティングアドバース (routingadvice)
1 6 7	シールドクラスフィーチャ (sealedClassFeature)
1 6 8	シールドインスタンスフィーチャ (sealedInstanceFeature)
1 6 9	セットメソッズ (setMethods)
1 7 0	セコンド (second)
1 7 1	サイズ (size)
1 7 2	サブジェクト (subject)
1 7 3	サブジェクトクラス (subjectClass)
1 7 4	サブジェクトネーム (subjectName)
1 7 5	サブジェクトノーツ (subjectNotes)

【2 3 4 3】

【表 2 2】

表A. 1 1	
1 7 6	スーパークラシズ (superClasses)
1 7 7	テレフォン (telephone)
1 7 8	チケットスタブ (ticketStub)
1 7 9	タイトル (titol)
1 8 0	トウメソッズ (toMethods)
1 8 1	トラベルノーツ (travelNotes)
1 8 2	バリュー (value)
1 8 3	バリアブルズ (valables)
1 8 4	ボキャブラリ (vocabulary)
1 8 5	ウェイ (way)
1 8 6	イヤー (year)
1 8 7	ゾーン (zone)

注：“イズインスタンス (isInstance)” は、オペレーションでもあるので、属性としてはリストされていない。

当てる。

【2 3 4 5】

【表 2 3】

【2 3 4 4】表A. 1 2は通路の形式へのコードを割り

表 A. 1 2	
1 9 1	バイコピー (byCopy)
1 9 2	保護された参照 (byProtectedRef)
1 9 3	参照 (byRef)
1 9 4	非保護された参照 (byUnprotectedRef)

5. 4. 3 エグゼキューティドオブジェクトエンコーディング (Execute Object Encodings) 【2 3 4 6】  
 表 A. 1 3 は、汎用の符号化に対するコードを割り当て 【表 2 4】

表 A. 1 3	
1	—
2	ビットストリング (BitString)
3	—
4	ゼネラルキャラクタ (GeneralCharacter)
5	アイデンティファイヤ (identifier)
6	ゼネラルインテジャ (GeneralInteger)
7	マーク (mark)
8	—
9	ニル (Nil)
1 0	オクテット (Octet)
1 1	オクテットストリング (OctetString)
1 2	プロシージャ (Procedure)
1 3	クオリファインドアイデンティファイヤ (QualifiedIdentifier)
1 4	リアル (Real)
1 5	—
1 6	ストリング (String)

注：これらのタグは、対応する予め規定されたクラスに対するコードである。 【2 3 4 8】  
 【2 3 4 7】表 A. 1 4 は、基本的な性質の特殊な目的 【表 2 5】

表 A. 1 4	
0	コメント (Comment)
— 1	ビットゼロ (BitZero)
— 2	ビットワン (BitOne)
— 3	ブーリアンフォルス (BooleanFalse)
— 4	スペシャルキャラクタ (SpecialCharacter)
— 5	ブーリアントルー (BooleanTrue)
— 6	プリ定義クラスアイデンティファイヤ (PredefinedClassIdentifier)
— 7	プリ定義フィーチャアイデンティファイヤ (PredefinedFeatureIdentifier)
— 8	インテジャマイナスイワン (IntegerMinusOne)
— 9	インテジャゼロ (IntegerZero)
— 1 0	インテジャプラスワン (IntegerPlusOne)

表 A. 1 5 は、モディファイヤの特別な目的の符号化へのコードを与える。 【2 3 4 9】  
 【表 2 6】



表 A. 15	
- 1 2	モディファイヤデマーカー (ModifierDemarcate)
- 1 3	モディファイヤークラス (ModifierGetClass)
- 1 4	モディファイヤプロパティ (ModifierGetProperty)
- 1 5	モディファイヤ変数 (ModifierGetVariable)
- 1 6	モディファイヤメンション (ModifierMention)
- 1 7	モディファイヤ属性 (ModifierSetAttribute)
- 1 8	モディファイヤプロパティ (ModifierSetProperty)
- 1 9	モディファイヤ変数 (ModifierSetVariable)
- 2 0	モディファイヤユーザスタック (ModifierUserStack)

表 A. 16 は、セレクターの特別な目的の符号化のためのコードを与える。

【2350】

【表27】

表 A. 16	
- 2 1	セレクタブレイク (SelectorBreak)
- 2 2	セレクタクライアント (SelectorClient)
- 2 3	セレクタコンティニュー (SelectorContinue)
- 2 4	セレクタエスカレート (SelectorEscalate)
- 2 5	セレクタブレイス (SelectorPlace)
- 2 6	セレクタプロセス (SelectorProcess)
- 2 7	セレクタセルフ (SelectorSelf)
- 2 8	セレクタサクシード (SelectorSucceed)

6. モジュールのシンタックス (Syntax of Module)  
ここに用いられている "module" は、1 以上のインタフェースを符号化するストリング、これらのインタフェースがその一部分であるクラスの識別子ならびに、最後に、モジュール自身の識別子である。

【2351】 "Identifier" 定義用語は、セクション6全体を通じて、全体としてのインスタンスではなく、クラス "Identifier" の1つのテキストを暗黙の内に参照する。

【2352】 注：以下に説明するように構成されたモジュールは、ハイテレスクリプトのモジュールでもある。この表現を偽とするいかなる下記の規則もドキュメンテーションの誤りである。

【2353】 6. 1 一般的な構造

モジュールは、以下に述べるシンタックティックルール及びそれに付随するセマンティックルールに従う一連のトークンである。BNFで表した場合、これらの規則は、オプションなトークンをブラケットで囲んでいる (" [" 及び " ] ")。さらに、モジュールは、キャラクタテレスクリプトにおいてパーミットされた種類のコメントも含みうる。これらのコメントに対する規定は、モジュールを支配するシンタックティックルールに含まれると考えられる。

【2354】 各々のトークンは1以上のキャラクタであ

る。モジュールは、トークンに従ってキャラクタを連鎖化することによって、そしてキャラクタテレスクリプトによってパーミットされた種類のブレイクキャラクタをトークンとの間に挿入することによって得られる。

【2355】 キーワード例えば "module" は、ケース (大文字、小文字) に感知する。アッパーケース (大文字) キャラクタは、以下に一般にモジュールを記述するために用いられ、ローアーケース (小文字) キャラクタは、このアペンディックス全体及びセクション7を通じてモジュールを作成するために用いられる。

【2356】 6. 2 詳細な構造

モジュールは次の詳細な構造を表す。

【2357】 6. 2. 1 モジュール (Module)

Module

モジュールは次の規則に従う。

【2358】 Module ::= identifier:MODULE=

(Interfaces)

Interface ::= identifier:Interface;

[Interfaces]

モジュールはそのモジュールを開始させる識別子によって表され、インタフェースを符号化する。これらの各々は、その識別子が識別子の直前にあるクラスのインタフェースである。

【2359】 どのインタフェースも、インタフェースの

特別のクラスのメンバに所属するオペレーション” initialize”を規定することができる。さらに何も述べられていなくても、ある定義が内在されている。すなわち、問題のクラスがミックスインであれば、” initialize: op 0 ;”であり、さもなければクラスのインタフェーススーパークラスの中のフレーバの定義である。

【2360】6. 2. 2 インタフェース (Interface)

Interface

インタフェースは次の規則に従う。

【2361】

```
Interface ::= [SEALED | ABSTRACT] INTERFACE
  ["FormalParameters"]
  [Superclasses] = ( [Definitions])
FormalParameters ::= ParameterGroup[;
FormalParamterGroup ::= Identifiers : CLASS
Superclasses ::= ( [ClassSpecifications] )
Definitions ::= DefinitionGroup ; [Definitions]
DefinitionGroup ::= [Access] [Responder]
Identifiers : Definition
Access ::= PUBLIC | PRIVATE | SYSTEM
Responder ::= INSTANCE | CLASS
Identifier ::= identifier [ Identifiers]
Definition ::= SEALED | [SEALED | ABSTRACT]
Feature
```

インタフェースの符号化はパラメータ化することができる。この符号化において、各々の識別子である1以上の” formal class parameters” が導入され、インタフェースがクラス識別子に要求する多くのプレースにおいて使用することができる（下記のトークン” ClassSpecifier” 参照）。インタフェースは、クラス” object” の識別子とその代わりに用いられたかのようなものである。同じように定義された公式のクラスパラメータは、その導入部分においてグループ化される。

【2362】各々識別子によって表されたゼロまたはそれ以上のフィーチャはシールされるかまたは定義される。同じようにシールされるかまたは定義されたフィーチャの識別子はグループ化される。第1のそのようなグループに、” Access” または” Responder” が存在しない場合、” PRIVATE” または” INSTANCE” はそれぞれそこに存在しているものと見做される。” Access” または” Responder” がどれかの後のグループに存在しない場合、前のグループに存在するキーワードは、次のグループに存在すると見做される。

【2363】インタフェースは次の通りである。属性” classFeatures” または” instanceFeatures” は、それぞれ” CLASS” または” INSTANCE” についてグループに定義されたフィーチャを含む。属性” isAbstract” は、” ABSTRACT” が存在している場合にのみ” true” である。属性” sealedClassFeatures” または” sealedIns

tanceFeatures” は、それぞれ” SEALED” 、” CLASS” または” INSTANCE” とグループを成している識別子を含む。属性” superclasses” は、トークン” Superclasses” が存在してれば、このトークン” Superclasses” が定める識別子であり、さもなければクラス” Object” のみの識別子である。クラスの属性中の位置は、そのスペシファイヤが左から右に移るに従って減少する。属性” vocabulary” はクリアされる。

【2364】インタフェースは、” SEALED” が存在している場合にのみシールされる。

【2365】6. 2. 3 フィーチャ

Feature

フィーチャの定義は次の規則に従う。

```
【2366】Feature ::= Attribute | Operation
Attribute
```

属性の定義は次の規則に従う。

```
【2367】Attribute ::= [READONLY] Constraint
[THROWS Identifiers]
```

属性の定義は次の通りである。属性” constraint” は与えられたコンストレイントである。属性” exceptions” は、” THROWS” が存在していれば、存在する識別子を含み、さもなければいかなる識別子も含まない。属性” isPublic” は、属性の定義を含む定義グループのためのキーが” PUBLIC” または” CLASS” である場合にのみ” true” である。属性” isSet” は” READONLY” が存在しない場合にのみ真である。

【2368】属性の定義の上記のモジュール符号化はそれが定義する属性の署名とともに言われる。

【2369】オペレーション

オペレーションの定義は次の規則に従う。

```
【2370】Operation ::= [UNPROTECTED] op (
ArgumentsAttribute)
```

```
[Constraint] [THROWS Identifiers]
```

定義オペレーションは次の通りである。属性” arguments” は示された通りである。属性” exceptions” は、” THROWS” が存在してれば存在している識別子であり、さもなければ識別子を含まない。しかし、” UNPROTECTED” が存在していれば、属性は、クラス” Reference Protected” の識別子をさらに付加的に表す。これは、レスポンドを変更するオペレーションがレスポンドの保護されたリファレンスを使用してリクエストされた場合に投出される。属性” isPublic” は、定義オペレーションを含む定義グループが” PUBLIC” または” CLASS” （上記のトークン” Interface” 参照）である場合にのみ” true” である。属性” result” は存在していればコンストレイントでありさもなければゼロである。

【2371】定義オペレーションの前述したモジュールの符号化は、定義オペレーションが定義するオペレーションの署名としばしば呼ばれる。

【2372】ArgumentsAttribute属性” arguments” は

次の規則に従う。

【2373】

ArgumentsAttribute ::= [Arguments] [···]

Arguments ::= ArgumentGroup [; Arguments]

ArgumentGroup ::= Identifiers : Constraint

ゼロまたはそれ以外のアーギュメントはそれらのものの  
コンストレイントによって記述される。各々のそのよう  
なアーギュメントは、識別子によって表される。同じよ  
うに拘束されるアーギュメントの識別子はグループ分け  
される。1以上のアーギュメントグループと省略符号(···)  
の両方が存在していれば、最後のグループは唯1つ  
のアーギュメントを含むものとされるが、ゼロまたはそ  
れ以上のアーギュメントを含むものと見做される。

【2374】属性"arguments"は、省略符号(···)が存  
在していなければ、ゼロまたはそれ以上の記述されたア  
ーギュメントに対するコンストレイントのリストであ  
り、さもなければゼロである。記述されるアーギュメ  
ントへのコンストレイントのリスト中の位置は、それら  
のものの識別子が左から右に移動するにつれて増大する。

【2375】6. 2. 4 コンストレイント

Constraint

コンストレイントは次の規則に従う。

【2376】Constraint ::= [COPIED | PROTECTED |  
UNPROTECTED] ClassSpecifier ["|"

NIL]

コンストレイントは次の通りである。属性"classId"  
は、トークン"ClassSpecifier"が定める識別子であ  
る。属性"isInstance"は、感嘆符("!")がトーク  
ン"ClassSpecifier"(下記参照)とともに存在してい  
る場合にのみ"true"である。属性"isOptional"  
は、"NIL"が存在している場合にのみ"true"であ  
る。属性"ofClass"はゼロである。属性"passage"  
は、"COPIED"、"PROTECTED"、"UNPROTECTED"また  
はこれらのどれもが存在していない時に、"byCop  
y"、"byProtectedRef"、"byUnprotectedRef"また

```

Telescript:  module=(
    Agent:  abstract interface (Process) = (
        private
        go: unprotected op (ticket: copied
            Ticket) TicketStub
        throws PermitViolated,
            StateImproper, TripException;
        send:unprotected op (tickets:
            copied List [Ticket])
            TicketStub|Nil
        thows PremitViolated,
            StateImproper, TripException;
    );
    Association:  sealed interface [keyClass, valueClass:
        Class] (Object, Odered) = (

```

は"byRef"となる。

【2377】ClassSpecifier

クラススペシファイヤは次の規則に従う。

【2378】ClassSpecifier ::= identifier [!]

["[" ClassSpecifiers "]" ]]

ClassSpecifiers ::= ClassSpecifier [

ClassSpecifiers]

識別子は、公式のプラスのパラメータであるかまたはク  
ラス"C"の識別子である。後者の場合、"C"のイン  
タフェースの符号化"E"がパラメータされた場合にの  
み、("["及び"]")の間に、"C"が有する公式  
のクラスパラメータと同数の実際のクラスパラメータ  
(各々のクラススペシファイヤである)が後続する。"  
C"のこの特別の使用は、"E"中の公式のクラスパ  
ラメータの各々の使用が対応して位置された実際のクラス  
パラメータの使用であるかのように行なわれる。

【2379】6. 2. 5 他非ターミナル

identifier

クラス"Identifier"の1つのインスタンスのテキスト  
として拘束されたキャラクタ。

【2380】7 プレデファインドモジュール

予め定義されたクラスのインタフェースは、以下に詳細  
に示されこのアペンディックスを通じて断片的に示され  
るモジュールによって規定される。命令セットの内部フ  
ィーチャはモジュールに表れない。

【2381】参照の目的のために用意されたこのセクシ  
ョンすなわちセクション7は、このアペンディックスの  
他の場所に見られる記述と重複する。

【2382】注：予め定義されたインタフェースの定義  
は、ハイテレスクリプトモジュールを形成する。これを  
可能にするために、いくつかのローテレスクリプト識別  
子は、"\_x"が後置されることによって、ハイテレス  
クリプトにおいて留保されたワードであるアイデンティ  
ファイヤを避けるようにする。

【2383】

```

        public
            key: keyClass;
            value: valueClass;
        system
            initialize: unprotected op (
                key: keyClass;
                value: valueClass);
    );
Attribute: sealed interface (Feature) = (
    public
        constraint: Constraint;
        isSet: Boolean;
    system
        initialize: unprotected op (
            constraint: Constraint|Nil;
            isSet, isPublic: Boolean|Nil;
            exceptions: set
[Identifier!] |Nil);
    );
Authenticator: abstract interface = 0;
    Bit: sealed interface (Primitive, Ordered) =
0;
    BitString: sealed interface (ConstrainedList
        [Bit], Executed) = (
        public
            constraint: sealed;
        system
            initialize: unprotected op
                (segments; Object ...
                /* Bit|protected BitString! */);
    );
    Boolean: sealed interface (Primitive, Ordered) =
        ( public
            and, or: op (boolean: Boolean)
Boolean;
            not: op 0 Boolean;
        );
    CalendarTime: interface = (
        public
            day, dst, hour, minute, month,
            second, year, zone:
            Integer|Nil;
            dayOfWeek, dayOfYear, readonly
            Integer|Nil;
            globalize, localize: unprotected op 0;
            normalize: unprotected op 0
Boolean;
        );
    Cased: abstract interface 0 = (
        public

```

```

        isLower, isUpper: abstract
        readonly Boolean;
        makeLower, makeUpper: abstract op 0 copied
        Cased;
    );
Character: sealed interface (Primitive, Cased,
    Oedered) = 0;
Citation: interface (object, Oedered) = (
    public
        author: Telename|Nil;
        majorEdition, minorEdition:
Integer|Nil;
        title: Identifier!;
    system
        initialize: unprotected op (
            title: Identifier!;
            author: Telename|Nil;
            majorEdition, minorEdition:
            Integer|Nil);
    );
Cited: abstract interface 0 = (
    public
        citation: readonly protected
Citation;
    system
        initialize: unprotected op (
            title: Identifier!;
            majorEdition, minorEdition:
            Integer);
    );
Class: sealed interface (Object,
    Cited, interchanged) = (
    public
        convert: sealed op (source:
            protected Object)
            copied Object
            throws ConversionUnavailable;
        isInstance: sealed op (instance_x:
            protected Object) Boolean;
        isMember: sealed op (member:
            protected Object) Boolean;
        isSubclass: sealed op (subclass:
            Class) Boolean;
        new: sealed op (parameters: Object ...) Object
            throws ClassAbstract, Exception,
            ObjectionUnitialized;
    system
        initialize: unprotected op 0
            throws FeatureUnavailable;
    );

```

```

ClassDefinition: sealed interface = (
    public
    implementation:
Implementation|Nil;
    interface_x: Interface;
    majorEdition, minorEdition:
Integer;
    makeClasses: op (definitions:
        protected ClassDefinition ...)
        Lexicon [Class]
    throws ClassException;
    title: Identifier!;
system
    initialize: unprotected op (title:
        Identifier!;
        majorEdition,
minorEdition: Integer;
        interface_x: Interface;
        implementation:
Implementation|Nil);
    );
    ClassException: abstract interface
(ProgrammException) = 0;
        ClassSealed: interface (ClassException)=0;
        ClassUndefined: interface
(ClassException)=0;
        FeatureRedefined:
            interface (ClassException) = 0;
        FeatureSealed: interface
            (ClassException) = 0;
        FeatureUndefined:
            interface (ClassException) = 0;
        MixinDisallowed:
            interface (ClassException) = 0;
        SuperclassesInvalid:
            interface (ClassException) = 0;
    Collection: interface [itemClass: Class] = (
        public
        clear: unprotected op 0;
        examine: op (item: protected
            itemClass) itemClass|Nil;
        exclude: unprotected op (item:
            protected itemClass)
            itemClass|Nil;
        include: unprotected op (item:
itemClass)
            throws ItemInvalid;
            length: readonly Integer;
            stream: op 0 copied Stream
                [itemClass];

```

```

        system
            initialize: unprotected op (items:
                itemClass ...)
            throws ItemInvalid;
    );
    Collection abstract interface
(ProgrammngException) = 0 ;
    Exception: ItemDuplicated:
        interface (CollectionException) = 0 ;
    ItemInvalid:
        interface (CollectionException) = 0 ;
    KeyDuplicated:
        interface (CollectionException) = 0 ;
    KeyInvalid:
        interface (CollectionException) = 0 ;
    ObjectsUnpaired:
        interface (CollectionException) = 0 ;
    PositionInvalid:
        interface (CollectionException) = 0 ;
    StackDepleted:
        interface (CollectionException) = 0 ;
    Constrained: abstract interface 0 = (
        public
        constraint:
        readonly protected Constraint;
        system
            initialize: unprotected op (
                constraint: copied Constraint|Nil);
    );
    Constrained interface [keyClass, valuClass: Class]
    Dictioncary: (Dictionary [keyClass, valuClass],
        Constrained)= (
        system
            initialize: unprotected op (
                constraint: copied Constraint;
                keysAndValues: Object ...
                /* key: keyClass; value: valueClass */
                throws keyDuplicated, KeyInvalid,
                ObjectsUnpaired;
    );
    ConstrainedList: interface [itemClass: Class] (List
        [itemClass], Constrained) = (
        system
            initialize: unprotected op (
                constraint: copied Constraint;
                items: itemClass ...)
            throws ItemInvalid;
    );
    ConstrainedSet: interface [itemClass: Class]
        (Set [itemClass], Constrained) = (

```

```

    system
        initialize: unprotected op (
            constraint: copied Constraint;
            items: itemClass ...)
        throws ItemInvalid;
    ItemInvalid;
);
Constraint: interface = (
    public
        classed: Identifier!;
        isInstance, isOptional: Boolean;
        ofClass: readonly Class|Nil;
        passege: Identifier!;
        throws PassageInvalid;
    system
        initialize: unprotected op (
            classId, passage:
Identifier!|Nil;
            isOptional, isInstance:
Boolean|Nil)
        throws PassageInvalid;
);
Contact: interface = (
    public
        subject: Process|Nil;
        subjectClass: readonly protected
            Citation|Nil;
        subjectName: readonly protected
            Telename|Nil;
        subjectNotes: Object|Nil;
    system
        initialize: unprotected op (
            subject: Process|Nil;
            subjectNotes: Object|Nil);
);
Contacted: abstract interface 0 = (
    private
        contacts: readonly Set [Contact];
);
Dictionary: interface [keyClass, ValueClass: Class]
    (Set [Association [keyClass,
        valueClass]]) = (
    public
        add: unprotected op (key: keyClass;
            value: valueClass)
        throws KeyInvalid;
        drop: unprotected op (key: protected
            keyClass) valueClass
        throws KeyInvalid;
        find: op (value: protected

```



```

        valueClass) keyClass|Nil;
get: op (key: protected keyClass)
    valueClass
throws keyInvalid;
rekey: unprotected op (
    currentkey: protected keyClass;
    newKey: keyClass)
throws keyInvalid;
set: unprotected op (
    key: protected keyClass;
    value: valueClass)
throws keyInvalid;
transpose: unprotected op (key1,
    key2: protected keyClass)
throws keyInvalid;
system
    initialize: unprotected op
        (keysAndValues: Object ...)
/* key: keyClass; value: valueClass */
throws KeyDuplicated, KeyInvalid,
    ObjectsUnpaired;
);
Exception: abstract interface (Objects, Unchanged) =
    (public
        throw_x: sealed op 0;
    );
Executed: abstract interface 0 = (
    public
        catch_x: sealed op (exception:
            Class) Exception|Nil
        throws Exception;
        do_x, loop_x: sealed op 0
        throws Exception;
        either: sealed op (false_x:
            Executed; precondition: Boolean)
        throws Exception;
        if_x: sealed op (precondition:
            Boolean)
        throws Exception;
        repeat_x: sealed op (repetitions:
            Integer)
        throws Exception;
        while_x: sealed op (precondition:
            Executed)
        throws Exception, ResultInvalid,
            ResultMissing;
    );
Execution abstract interface (KernelException) =
0;
Exception:

```

```

ArgumentInvalid:
    interface (ExecutionException) = 0 ;
ArgumentMissing:
    interface (ExecutionException) = 0 ;
AttributeReadOnly: interface
    (ExecutionException) = 0 ;
ClassUnavailable:
    interface (ExecutionException) = 0 ;
EscalationInvalid:
    interface (ExecutionException) = 0 ;
FeatureUnavailable:
    interface (ExecutionException) = 0 ;
InternalException:
    interface (ExecutionException) = 0 ;
Property Undefined:
    interface (ExecutionException) = 0 ;
ReferenceProtected:
    interface (ExecutionException) = 0 ;
ReferenceVoid:
    interface (ExecutionException) = 0 ;
ResponderMissing:
    interface (ExecutionException) = 0 ;
ResultInvalid:
    interface (ExecutionException) = 0 ;
ResultMissing:
    interface (ExecutionException) = 0 ;
VariableUndefined:
    interface (ExecutionException) = 0 ;
Feature: abstract interface = (
    public
        exceptions: Set [Identifier!];
        isPublic: Boolean;
    system
        initialize: unprotected op (
            isPublic: Boolean[Nil];
            exception:Set
[Identifier!] [Nil]);
    );
Hased: abstract interface 0 = (
    public
        hash: abstract readonly Interger;
    );
Identifier: sealed interface (Primitive, Ordered) =
0 ;
Implementation: sealed interface = (
    public
        classMethods, fromMethods,
        instanceMethods, setMethods,
        toMethods: Lexicon [Method];
        properties: List [Idenntifier!];

```

```

    superclasses: List
    [Identifier!] | Nil;
    vocabulary: Lexicon
    [Citation] | Nil;
system
    initilize: unprotected op (
        superclasses: List
        [Identifier!] | Nil;
        vocabulary: Lexicon
        [Citation] | Nil;
        properties: List
        [Identifier!] | Nil;
        instanceMethods, setMethods,
        classMethods,
        formMethods, toMethods: Lexicon
        [Methods] | Nil);
);
Integer: sealed interface (Number) = (
    public
        modulus, quotient: op (divisor:
            Integer) Integer
        throws DivisionByZero;
);
Interchanged: abstract interface (Unchanged) = (
    public
        digest: abstract readonly
        protected Object | Nil;
);
Interface: sealed interface = (
    public
        classFeatures, instanceFeatures:
            Lexicon [Feature];
        isAbstract: Boolean;
        sealedClassFeatures,
        sealedInstanceFeatures:
            Set [Identifier!];
        superclasses: List [Identifier!];
        vocabulary: Lexicon [Citation];
system
    initialize: unprotected op (
        superclasses: List
        [Identifier!] | Nil;
        vocabulary: Lexicon
        [Citation] | Nil;
        instanceFeatures:
            Lexicon [Feature] | Nil;
        sealedInstanceFeatures:
            Set [Identifier!] | Nil;
        classFeatures: Lexicon
        [Feature] | Nil;

```

```

sealedclassFeatures:
    Set [Identifier!] | Nil;
    isAbstract: Boolean | Nil;
);

Kernel
Exception: abstract interface
    (ProgrammingException) = 0;
Classabstract: interface
    (KernelException) = 0;
ConversionUnavailable:
    interface (KernelException) = 0;
CopyUnavailable:
    interface (KernelException) = 0;
LoopMissing: interface (KernelException)
    = 0;
MakeMissing: interface (KernelException)
    = 0;
ObjectUninitialized:
    interface (KernelException) = 0;
PassageInvalid:
    interface (KernelException) = 0;
SelectorDuplicated:
    interface (KernelException) = 0;
Lexicon: interface [valueClass: Class]
    (ConstrainedDictionary [Identifier,
        valueClass]) = (
    public
        constraint: sealed;
    system
        initialize: unprotected op
            (keysAndValues: Object ...
            /* key: Identifier!; value:
            valueClass */)
        throws KeyDuplicated, KeyInvalid,
            ObjectsUnpaired;
);
List: interface [itemClass: Class]
    (Collection [itemClass], Ordered)
    = (
    public
        add: unprotected op (position:
            Integer; item: itemClass) throws
            ItemInvalid, PositionInvalid;
        drop: unprotected op (position:
            Integer) itemClass
        throws positionInvalid;
        find: op (
            initialPosition: Integer;
            item: protected itemClass)
            Integer | Nil

```

```

        throws PositionInvalid;
        get: op (position: Integer)
        itemClass
        throws PositionInvalid;
        reposition: unprotected op
            (currentPosition, newPosition:
            Integer)
        throws PositionInvalid;
        set: unprotected op (position:
        Integer; item: itemClass) throws
        ItemInvalid, PositionInvalid;
        transpose: unprotected op
            (position1, position2: Integer)
        throws PositionInvalid;
    system
        initialize: unprotected op (items:
            itemClass ...);
    );
    Mark: sealed interface (Primitive) = 0;
    Means: abstract interface = 0;
    MeetingException: abstract interface (Exception) = 0;
    MeetingDenied:
        interface (MeetingException) = 0;
    MeetingDuplicated:
        interface (MeetingException) = 0;
    MeetingInvalid:
        interface (MeetingException) = 0;
    PetitiononExpried:
        interface (MeetingException) = 0;
    MeetingPlace: abstract interface (Place) = (
        public
        meet: unprotected op (petition:
            copied Petition) Contact
        throws MeetingException,
            ProcessNotCurrent,
            StateImproper;
        part: unprotected op (contact:
            Contact)
        throws MeetingInvalid,
            ProcessNotCurrent,
            StateImproper;
        partAll: unprotected op ()
        throws ProcessNotCurrent,
            StateImproper;
    );
    Method: sealed interface = (
        public
        procedure: Procedure;
        variables: List [Identifier!];
    system

```

```

        initialize: unprotected op (
        procedure: Procedure|Nil;
        variables: List
[Identifier!]|Nil);
    );
    Miscellaneous abstract interface
        (ProgrammingException) = 0;
    Exception: PatternInvalid: interface
        (MiscellaneousException) = 0;
    SeedInvalid:
        interface (MiscellaneousException) =
0;
    Modifier: sealed interface (Primitive) = 0;
    Named: abstract interface 0 = (
        public
        name: sealed readonly protected
Telename;
    );
    Nil: sealed interface (Primitive) = 0;
    Number: abstract interface (Primitive, Ordered
    = (
        public
        abs, negate: abstract op 0
Number;

        add, multiply: abstract op
        (number: Number) Number;
        ceiling, floor, round, truncate:
        abstract op 0 Integer;
        divide: abstract op (divisor:
        Number) Number
        throws DivisionByZero;
        sybtract: abstract op (subtrahend:
        Number) Number;
    );
    Object: abstract interface (Referenced) = (
        public
        class: sealed readonly Class;
        copy: sealed op 0 copied Object
        throws CopyUnavailable;
        isEqual: op (object: protected
        Object) Boolean;
        select: sealed op (association:
        Object ...
        /* protected Object; Executed */
        throws Exception,
        SelectorDuplicated;
        size: sealed readonly Integer;
    system
        finalize, initialize: unprotected
        op 0;

```

```

    );
    Octet: sealed interface (Primitive, Ordered) =
0;
    OctetString: sealed interface (ConstrainedList
        [Octet], Excuted) = (
        public
        constraint: sealed;
        system
        initialize: unprotected op
            (segments: Object ...
            /* Octet|protected OctetString!
            */);
    );
    Operation: sealed interface (Feature) = (
        public
        arguments: List [Constraint] |Nil;
        result: Constraint|Nil;
        system
        initialize: unprotected op(
            arguments: List
[Constraint] |Nil;

            result: Constraint|Nil;
            isPublic: Boolean|Nil;
            exceptions: Set
[Identifier] |Nil);
    );
    Ordered: abstract interface () = (
        public
        isAfter, isBefore: abstract op
            (object: protected Ordered)
            Boolean;
        max, min: op (object: Ordered)
            Ordered;
    );
    Package: interface (ConstrainedSet [Class],
        Cited, Interchanged) = (
        public
        constraint: sealed;
        system
        initialize: unprotected op (
            title: Identifier!;
            majorEdition, minorEdition:
            Integer;
            items: Class ...)
            throws ItemDuplicated,
            ProcessNotPeer;
    );
    Pattern: interface (Object, Ordered) = (
        public
        find: op (string: protected

```

```

        String!; position: Integer|Nil)
        List[Integer]|Nil)
        throws PositionInvalid;
    substitute: op (
        repetitions: Integer;
        string: unprotected String!;
        replacement: protected String!)
        Integer;
system
    initialize: unprotected op (text:
        copied String!)
        throws PatternInvalid;
);
Permit: interface (Object, Ordered) = (
    public
        age, authenticity, charges,
        extent, priority: Integer|Nil;
        canChange, canCreate, canDeny,
        canGo, canGrant, canRestart,
        canSend: Boolean;
        intersect: op (permit: Permit)
            Permit;
    system
        initialize: unprotected op (age,
            charges, extent: Integer|Nil);
Petition: interface = (
    public
        agentClass: Citation|Nil;
        agentName: Telname|Nil;
        maximumWait: Integer|Nil;
    system
        initialize: unprotected op (
            agentName: Telename|Nil;
            agentClass: Citation|Nil;
            maximumWait: Integer|Nil;
);
Petitioned: abstract interface () = (
    system
        meeting: unprotected op (
            contact: Contact; petition:
                protected Petition)
            throws MeetingDenied;
        parting: unprotected op (contact:
            Contact);
);
Place: abstract interface (Process, Unmoved) = (
    public
        address: sealed readonly protected
            Teleaddress;
        publicClasses: sealed readonly Set

```



```

        [Cited];
        terminate: sealed unprotected op (
            occupant: protected Telename)
            Boolean
        throws ProcessNotCurrent,
            ProcessNotPeer, StateImproper;
system
    entering: unprotected op (
        contact: Contact;
        permit: protected Permit;
        ticket: protected Ticket|Nil)
    throws OccupancyDenied;
    exiting: unprotected op (
        contact: Contact;
        permit: protected Permit;
        ticket: protected Ticket|Nil);
    );
Primitive: abstract interface (Object, Excuted,
    Unchanged) = (
    system
        initialize: unprotected op ()
        throws FeatureUnavailable;
    );
Primitive abstract interface
(ProgrammngException) = ();
Exception:
    DivisionByZero: interface
        (PrimitiveException) = ();
Procedure: sealed interface (Primitive) = ();
Process: abstract interface (Object, Named,
    Uncopied)=(
    public
        brand: sealed readonly protected
        Object
        throws ProcessNotPeer;
        localPermit: sealed copied permit
        throws FeatureUnavailable,
            PermitVoilated;
        nativePermit: sealed copied Permit
        throws FeatureUnavailable,
            PermitVoilated;
        permit: sealed readonly copied
        Permit;
        throws ProcessNotPeer;
        regionalPermit: sealed copied
        Permit
        throws FeatureUnavailable,
            PermitVoilated;
    private
        age: sealed readonly Integer;

```

```

charges: sealed readonly Integer;
contacts: sealed readonly

Set [Process] :

priority: sealed Integer|Nil;
privateClasses: sealed readonly
  Set [Cited] ;
public
charge: sealed op (charges:
  Integer)
throws PermitInadequate,
  PermitViolated;
restrict_x: sealed op (
  procedure: Procedure;
  permit: protected permit)
throws Exception, PermitViolated,
  ProcessNotCurrent;
sponsor_x: sealed op (
  procedure: Procedure;
  permit: protected permit)
throws Exception, PermitViolated,
  ProcessNotCurrent;
wait: unprotected op (seconds:
  Integer);
system
initialize: op (
  nativePermit: copied Permit;
  privateClasses: Set
    [Cited] |Nil)
throws PermitViolated;
live: abstract unprotected op
  (cause: Exception|Nil)
throws Exception;
restricted: op (permit:
  Identifier; isRelocated:
  Boolean) PermitReduced|Nil;
);
Process- abstract interface
  (ProgrammingException) = 0 ;
Exception:
  PermitInadequate: interface
    (ProcessException) = 0 ;
ConditionUnavailable:
  interface (ProcessException) = 0 ;
ConditionUndefined:
  interface (ProcessException) = 0 ;
PermitExhausted:
  interface (ProcessException) = 0 ;
PermitViolated:
  interface (ProcessException) = 0 ;
ProcessNotCurrent:

```

```

        interface (ProcessException) = 0 ;
ProcessNotPeer:
        interface (ProcessException) = 0 ;
ResourceUnavailable:
        interface (ProcessException) = 0 ;
StateImproper:
        interface (ProcessException) = 0 ;
Programming abstract interface (Exception) = 0 ;
Exception:
Qualified sealed interface (Identifier) = 0 ;
Identifier:
RandomStream: interface (Stream [Integer]) = (
    system
        initialize: unprotected op (seed:
            Integer)
        throws SeedInvalid;
    );
Real: sealed interface (Number) = 0 ;
Referenced: abstract interface () = (
    public
        discard: sealed op ();
        isProtected: sealed readonly
            Boolean;
        isSame: sealed op (reference:
            protected Referenced) Boolean;
        protect: sealed op () protected
            Referenced;
        ref_x: sealed unprotected op ();
    );
Resource: interface = (
    public
        condition: Identifier!
        throws ConditionUnavailable;
        conditions: readonly protected Set
            [Identifier!];
        use: sealed unprotected op(
            procedure: Procedure;
            exclusive: Boolean|Nil;
            maximumWait: Integer|Nil;
            conditions: copied Set
                [Identifier!]|Nil) Boolean
        throws ConditionUndefined,
            Exception, ResourceUnavailable;
    system
        initialize: unprotected op(
            condition: Identifier!|Nil;
            conditions: copied Set
                [Identifier!]|Nil) throws
            ConditionUndefined;
    );

```

```

Selector: sealed interface (Primitive) = 0;
Set: interface [itemClass: Class]
      (Collection [itemClass], Verified) =
      (
      public
        difference, intersection, union:
          unprotected op (set: protected Set
            [itemClass]);
      system
        initialize: unprotected op (items:
          itemClass ...)
        throws ItemDuplicated, ItemInvalid;
      );
Stack: interface [itemClass: Class] (List
      [itemClass]) = (
      public
        pop: unprotected op () itemClass
        throws StackDepleted;
        push: unprotected op (item:
          itemClass);
        pushItems: unprotected op (
          items: protected List
            [itemClass]);
        roll: unprotected op (shifts,
          items: Integer)
        throws ArgumentInvalid,
          StackDepleted;
        swap: unprotected op ()
        throws StackDepleted;
      );
Stream: abstract interface [itemClass: Class] =
      (
      public
        current: abstract readonly
          itemClass|Nil;
        isDone: abstract Boolean;
        next: abstract readonly
          itemClass|Nil
        throws ReferenceProtected;
      );
String: sealed interface
      (ConstrainedList [Character], Cased,
      Executed) = (
      public
        constraint: sealed;
        substring: op (initialPosition,
          beyondFinalPosition: Integer)
          copied String
        throws PositionInvalid;
      system

```

```

        initialize: unprotected op
            (segments: Object ...
            /* Character|protected String!
            */);
    );
Teleaddress: interface = (
    public
        location: String!|Nil;
        provider: OctetString!;
        routingAdvice: List
            [OctetString!];
    system
        initialize: unprotected op (
            provider: OctetString!|Nil;
            location: String!|Nil);
    );
Telename: interface = (
    public
        authority: OctetString!;
        identity: OctetString!|Nil;
    system
        initialize: unprotected op (
            authority, identity:
                OctetString!|Nil);
    );
Telenumbr: interface = (
    public
        coutry, telephone: String!;
        extension: String!|Nil;
    system
        initialize: unprotected op (
            countryAndTelephone: String!;
            extension: String!|Nil);
    );
Ticket: interface (TicketStub) = (
    public
        desiredWait, maximumWait:
            Integer|Nil;
        destinationAddress:
            Teleaddress|Nil;
        destinationClass: Citation|Nil;
        destinationName: Telename|Nil;
        destinationPermit: Permit|Nil;
    system
        initialize: unprotected op (
            destinationName: Telename|Nil;
            destinationAddress:
                Teleaddress|Nil;
            destinationClass: Citation|Nil;
            maximumWait: Integer|Nil;

```

```

        way: Way|Nil;
        travelNotes: Object|Nil);
    );
TicketStub: interface = (
    public
        travelNotes: Object|Nil);
        way: Way|Nil;
    system
        initialize: unprotected op (
            way: Way|Nil;
            travelNotes: Object|Nil);
    );
Time: interface (Object, Ordered, Unchanged) =
    (
        public
        adjust: op (seconds: Integer) Time;
        interval: op (subtrahend: Time) Integer;
    );
TripException: abstract interface (Exception) = (
    public
        ticketStub: sealed TicketStub;
    system
        initialize: unprotected op
            (ticketSub: TicketStub);
    );
DestinationUnavailable:
    interface (TripException) = 0;
DestinationUnknown:
    interface (TripException) = 0;
OccupancyDenied:
    interface (TripException) = 0;
TicketExpired: interface (TicketException)
    = 0;
WayUnavailable: interface
    (TripException) = 0;
Unchanged: abstract interface 0 = 0;
Unexpected: interface (ExecutionException) = (
Exception: public
    exception: readonly Exception;
    system
        initialize: unprotected op (
            exception: Exception);
    );
Unmoved: abstract interface 0 = 0;
Verified: abstract interface 0 = (
    public
        verify: abstract op 0 Boolean;
    );
Way: interface = (
    public

```

```

    authenticator: Authenticator|Nil;
    means: Means|Nil;
    name: Telename|Nil;
system
    initialize: unprotected op (
        name: Telename|Nil;
        means: Means|Nil;
        authenticator:
            Authenticator|Nil);
    );
) /*Telescript */

```

## 8 プレデファインドクラスグラフ

以下のダイアグラムは、予め定義されたクラスを含むクラスグラフの一部分を示している。あるクラスのイミューディエイトサブクラスはその直下にインデントされている。アンダーラインされたクラスは抽象的である。括弧（”（”及び”）”）はミックスインを囲んでいる。

【2384】オブジェクト（参照される）

- ・ アソシエーション（順序あり）
- ・ オーセンティケータ
- ・ カレンダ時間
- ・ サイテーション（順序あり）
- ・ クラス（引用され、相互変換される）
- ・ クラス定義
- ・ コレクション
- ・ リスト（順序あり）
- ・ コンストレインドリスト（コンストレインド）
- ・ ビットストリング（実行済み）
- ・ オクテットストリング（実行済み）
- ・ スtring（ケースされ、実行される）
- ・ スタック
- ・ セット（検査済み）
- ・ コンストレインドセット（コンストレインド）
- ・ パッケージ（引用され、相互変換される）
- ・ ディクショナリ
- ・ コンストレインドディクショナリ（コンストレインド）
- ・ レキシコン
- ・ コンストレイント
- ・ コンタクト
- ・ 例外（変更なし）
- ・ Meeting Exception
- ・ Programming Exception
- ・ Class Exception
- ・ Collection Exception
- ・ Kernel Exception
- ・ Execution Exception
- ・ Unexpected Exception
- ・ Miscellaneous Exception
- ・ Primitive Exception

- ・ Process Exception
- ・ Trip Exception
- ・ Feature
- ・ 属性
- ・ オペレーション
- ・ インプリメンテーション
- ・ インタフェース
- ・ Means
- ・ 方法
- ・ パタン（順序あり）
- ・ パーミット（順序あり）
- ・ ペティション
- ・ Primitive (Executed & Unchanged)
- ・ ビット（順序あり）
- ・ ブーリアン（順序あり）
- ・ キャラクタ (Cased & Ordered)
- ・ アイデンティファイヤ（順序あり）
- ・ 有資格識別子
- ・ マーク
- ・ モディファイヤ
- ・ ゼロ
- ・ Number (Ordered)
- ・ 整数
- ・ 実数
- ・ オクテット（順序あり）
- ・ プロジージャ
- ・ セレクター
- ・ Process (Named)
- ・ Agent
- ・ Place (Unmoved)
- ・ Meeting Place
- ・ リソース
- ・ Stream
- ・ ランダムストリーム
- ・ テレアドレス
- ・ テレネーム
- ・ テレナンバ
- ・ チケットスタブ
- ・ チケット

・ タイム (順序あり、変更なし)	3. 4. 7	スタック
・ ウェイ	3. 4. 8	ストリング
Cased	3. 5	リファレンスエンコーディング
Cited	3. 5. 1	ゼネラルリファレンス
Constrained	3. 5. 2	ボイディドリファレンス
Contacted	3. 5. 3	インターチェンジリファレンス
Executed	3. 5. 4	クラスリファレンス
Hashed	3. 5. 5	プロシージャリファレンス
Named	4.	エンコーディングクラスサイズと属性
Ordered	4. 1	コンベンションズ
Petitioned	4. 2	エンコーディンググループ
Referenced	4. 2. 1	キャッチフレーム
Unchanged	4. 2. 2	コレクションストリーム
・ Interchanged	4. 2. 3	ダイヤルストリング
Unmoved	4. 2. 4	フレーム
Verified	4. 2. 5	ゴーフレーム
アベンディックスB	4. 2. 6	プレデファインドフレーム
1. 序文	4. 2. 7	プロシージャフレーム
1. 1 構成	4. 2. 8	リピートフレーム
1. 2 参考文献	4. 2. 9	レストリクトフレーム
2 符号化の概念	4. 2. 10	センドフレーム
2. 1 テレパーセル	4. 2. 11	ユースフレーム
2. 2 オブジェクトエンコーディング	4. 2. 12	ユースデファインドフレーム
2. 2. 1 形式	4. 2. 13	ホワイルフレーム
2. 2. 2 規則	4. 3	エンコーディングアトリビュート
2. 2. 3 パレット	4. 3. 1	エージェント
2. 2. 4 エンコーディングクラス	4. 3. 2	キャッチフレーム
2. 2. 5 エンコーディングアトリビュート	4. 3. 3	コレクション
2. 3 アトリビュートエンコーディング	4. 3. 4	コレクションストリーム
2. 3. 1 形式	4. 3. 5	ディクショナリ
2. 3. 2 規則	4. 3. 6	リスト
2. 4 リファレンスエンコーディング	4. 3. 7	パターン
2. 4. 1 形式	4. 3. 8	プロシージャフレーム
2. 4. 2 規則	4. 3. 9	ランダムストリーム
3. エンコーディングスペシフィケーション	4. 3. 10	リピートフレーム
3. 1 コンバージョン	4. 3. 11	レストリクトフレーム
3. 2 テレパーセル	4. 3. 12	センドフレーム
3. 3 オブジェクトエンコーディング	4. 3. 13	スタック
3. 3. 1 エグゼキューティドオブジェクト	4. 3. 14	タイム
3. 3. 2 プレデファインドオブジェクト	4. 3. 15	ユースフレーム
3. 3. 3 ユーザデファインドオブジェクト	4. 3. 16	ユースデファインドフレーム
3. 3. 4 プロテクティドオブジェクト	4. 3. 17	ホワイルフレーム
3. 3. 5 パレット	5. 1	コンベンションズ
3. 4 アトリビュートエンコーディング	5. 2	ランゲージクラスサイズ
3. 4. 1 ブーリアン	5. 2. 1	エージェント
3. 4. 2 ダイヤルストリング	5. 2. 2	アソシエーション
3. 4. 3 整数	5. 2. 3	アトリビュート
3. 4. 4 アイテム	5. 2. 4	カレンダータイム
3. 4. 5 リスト	5. 2. 5	サイティション
3. 4. 6 オクテットストリング	5. 2. 6	クラス



- 5. 2. 7 クラスデフィニション
- 5. 2. 8 コレクション
- 5. 2. 9 コンストレインドディクショナリ
- 5. 2. 10 コンストレインドリスト
- 5. 2. 11 コンストレインドセット
- 5. 2. 12 コンストレイント
- 5. 2. 13 コンタクト
- 5. 2. 14 ディクショナリ
- 5. 2. 15 インプリメーション
- 5. 2. 16 インターフェース
- 5. 2. 17 レキシコン
- 5. 2. 18 リスト
- 5. 2. 19 メソッド
- 5. 2. 20 ミックスイン
- 5. 2. 21 オペレーション
- 5. 2. 22 パッケージ
- 5. 2. 23 パターン
- 5. 2. 24 パーミット
- 5. 2. 25 ペティション
- 5. 2. 26 ランダムストリート
- 5. 2. 27 リソース
- 5. 2. 28 セット
- 5. 2. 29 スタック
- 5. 2. 30 テレアドレス
- 5. 2. 31 テレネーム
- 5. 2. 32 テレナンバ
- 5. 2. 33 チケット
- 5. 2. 34 チケットタブ
- 5. 2. 35 タイム
- 5. 2. 36 トリップエグゼクション
- 5. 2. 37 アンエクスベクティドエクセプション
- 5. 2. 38 ウエイ
- 5. 3 エンコーディングクラシース
- 5. 3. 1 キャッチフレーム
- 5. 3. 2 コレクションストリーム
- 5. 3. 3 プレデファインドフレーム
- 5. 3. 4 リピートフレーム
- 5. 3. 5 レストリクトフレーム
- 5. 3. 6 センドフレーム
- 5. 3. 7 ユースフレーム
- 5. 3. 8 ユースデファレンドフレーム
- 5. 3. 9 ホワイルフレーム

#### 1. 序文

この開示のアペンディックスAのセクション1は、本発明の主要素を紹介し、その説明は引用によってこのアペンディックスの一部とされる。アペンディックスAのセクションの1. 4. 3に示されたコンベンションは、このアペンディックスでも同じように用いられている。

#### 【2385】1. 1 構成

このアペンディックスは3つのセクションに分けられ

る。セクション1はこの序である。セクション2は符号化規則の主なコンセプトを紹介する。このセクション3は命令セットで用いられているもの以上に、符号化において用いられている予め定義されたクラス及び属性を定義する。

#### 【2386】1. 2 参考文献

このアペンディックスは、次の別の文書に依存している [10646]

Information technology--Universal Coded Character Set (UCS), ISO/IEC DIS10646, International Organization for Standardization and International Electrotechnical Commission, 1990.

[Telescript]

この開示のアペンディックスA.

#### 【2387】[Unicode]

The Unicode Standard: orldwide Character Encoding, Volume 1, Version 1.0, The Unicode Consortium, Addison-Wesley, 1991.

#### 2 符号化の概念

テレパーセルはこのアペンディックスのこのセクションにおいて構想される。サブセクションは、オブジェクト、属性及び参照の符号化ではなくテレパーセルに向けられる。

#### 【2388】2. 1 テレパーセル (Teleparcel)

”テレパーセル (teleparcel)” は、テレパーセルの”サブジェクト (subject)” であるオブジェクト、サブジェクトの成分及びそれらのものの成分の全てを回帰的に符号化する。サブジェクトまたはコンポーネントのクラスは、予め定義されていてもユーザによって定義されていてもよい。

【2389】注：テレパーセルは、その標準的な形式においてのテレスクリプトオブジェクトである。

【2390】注：テレパーセルは、オブジェクトを2つのエンジンの間に搬送する手段である。ソースエンジンは、オブジェクトをテレパーセルに変換しそれを目的点のエンジンに搬送する。目的点のエンジンは、テレパーセルをオブジェクトに変換することによって、ソースにおいて消費されたものと同じオブジェクトを目的点において作り出す。

【2391】注：テレパーセルのサブジェクトは、典型的には排他的ではなくてもエージェントである。このような場合、テレパーセルは、エージェントと、エージェントが所有する全てのオブジェクトと、エージェントの現在の実行状態を形成するものを含めて包括する。

#### 【2392】2. 2 オブジェクトエンコーディング (Object Encoding)

”オブジェクトエンコーディング (object encoding)” はオブジェクトとそのオブジェクトへの参照を符号化する。

#### 【2393】2. 2. 1 形式

オブジェクトは、以下の形式の内のどれか1つを取る。

【2394】 Executed

”エグゼキューティッドオブジェクトエンコーディング (executed object encoding)” は実行されるオブジェクトを符号化する。

【2395】 Predefined

”プリディファインドオブジェクトエンコーディング (predefined object encoding)” は、そのクラスが予め定義された実行されていないオブジェクトを符号化する。

【2396】 User-defined

”ユーザーディファインドオブジェクトエンコーディング (user-defined object encoding)” は、そのクラスがユーザー定義されるオブジェクトを符号化する。

【2397】 Protected

”プロテクティッドオブジェクトエンコーディング (protected object encoding)” は、任意のオブジェクトを符号化する。

【2398】オブジェクトの符号化は、そのオブジェクトへの参照も符号化する。参照は、オブジェクトが参考されていないかまたはオブジェクトの符号化形式が保護されている場合にのみ保護される。

【2399】 2. 2. 2 規則

あるオブジェクト及び参照に対するオブジェクトの符号化の形式は、以下の規則に従って選択されるものとする。これらの規則はここで示した順序で適用される。

【2400】 1. 参照が保護され、オブジェクトが変更されない場合に、プロテクティッドオブジェクトエンコーディングが選択される。

【2401】 2. オブジェクトが実行される場合、実行されるオブジェクトの符号化が選択される。

【2402】 3. オブジェクトのクラスが予め定義されているならば、予め定義されたオブジェクトの符号化が選択される。

【2403】 4. (その他の場合には) ユーザによって定義されたオブジェクトの符号化が選択される。

【2404】 2. 2. 3 パレット

オブジェクトの符号化は、オブジェクトの符号化に含めることを支配する予め定義された属性及び規則のセットである”エンコーディングパレット”に依存することができる。

【2405】予め定義されたオブジェクトまたはユーザによって定義されたオブジェクトの符号化は、符号化されたオブジェクト”0”が1つの例である予め定義されたクラスによって定められた”1次”パレットに依存する。あるオブジェクトは、オブジェクトが予め定義されたクラスのインプリメンテーションメンバであるがその予め定義されたサブクラスのインプリメンテーションメンバではない場合にのみ、その予め定義されたクラスの”0”である。

【2406】ユーザによって定義されたオブジェクトの符号化は、”0”が1つの例である予め定義されたクラスのインスタンスが引用も、制約もまた名前付けもされていないのに、”0”が引用され制約されまた名前付けされている場合にのみ、”二次”パレット (このアペンディックスのセクション5. 2. 20に定める) に依存する。

【2407】あるパレットにおいて各々の予め定義された属性は、命令的または恣意的と表示される。”命令的 (optional)” 属性は、パレットに依存する符号化において表され、”恣意的な (mandatory)” 属性は表現されてもよいが必ずしもその必要はない。符号化の目的のためにある数を各々の恣意的な属性に割り当てる。

【2408】注：いかなるオブジェクトもミックスインではない。

【2409】 2. 2. 4 エンコーディングクラス (Encoding Classes)

エンコーディングパレットは、このアペンディックスのセクション4. 2に記述されたように符号化クラスを容認する。”符号化クラス (encoding class)” は、—この開示のアペンディックスAに記述されたクラスである”言語クラス (language classes)” を超えた—予め定義されたクラスであり、符号化の目的のために命令セットを指示するが、命令セットの一部ではない。

【2410】注：符号化クラスの唯一つの目的は、符号化特性を定義することである。このクラスはイニシャライゼーションパラメータ、オペレーション、アダプテーションまたは変更を持たない。

【2411】 2. 2. 5 エンコーディングアトリビュート (Encoding Attributes)

符号化パレットは、このアペンディックスのセクション4. 3に記述された符号化属性を容認する。”符号化属性 (encoding attribute)” は、—この開示のアペンディックスAに記述された”言語属性 (language attributes)” を超えた—予め定義された属性であり、符号化目的のための命令セットを指示するが、命令セットの一部ではない。各々の符号化属性は、システムインスタンス属性である。

【2412】注：符号化属性は、ユーザが定義したクラスのインタフェースが同一の識別子を属性に与えることを排除しない。

【2413】 2. 3 アトリビュートエンコーディング”属性の符号化 (attribute encoding)” は予め定義された属性とこの属性の参照を符号化する。

【2414】 2. 3. 1 形式

属性の符号化は次の形式のいずれかを取る。

【2415】 Boolean

”ブーリアン属性符号化 (boolean attribute encoding)” はクラス”ブーリアン (Boolean)” のインスタンスを符号化する。

## 【2416】Dial String

”ダイアルストリングアトリビュートエンコーディング (dial string attribute encoding)” はクラス”ダイアルストリング (Dial String)” のインスタンスを符号化する。

## 【2417】Integer

”インテジャアトリビュートエンコーディング (integer attribute encoding)” はクラス”インテジャ (Integer)” の1つの例を符号化する。

## 【2418】Item

”アイテム属性符号化 (Item attribute encoding)” は、オブジェクトまたはオブジェクトへの参照を符号化する。

## 【2419】List

”リストアトリビュートエンコーディング (list attribute encoding)” はクラス”リスト (List)” のインスタンスを符号化する。

## 【2420】Octet String

”オクテットストリングアトリビュートエンコーディング (octet string attribute encoding)” はクラス”オクテットストリング (Octet String)” の1つの例を符号化する。

## 【2421】Stack

”スタックアトリビュートエンコーディング (stack attribute encoding)” はクラス”スタック (Stack)” の1つの例を符号化する。

## 【2422】String

”ストリングアトリビュートエンコーディング (string attribute encoding)” はクラス”ストリング (String)” の1つのインスタンスを符号化する。

## 【2423】2. 3. 2 規則

属性の属性符号化の形式は、次の規則に従って選択され、これらの規則は次の順序で適用される。

【2424】1. アイテム属性以外の属性の符号化は、その属性が必要なクラスの1つのインスタンスである場合に選択される。

【2425】2. (他の場合には) アイテム属性符号化が選択される。

## 【2426】2. 4 リファレンスエンコーディング (Reference Encoding)

”リファレンスエンコーディング (reference encoding)” はオブジェクトの符号化によって符号化されるオブジェクトへのリファレンスを符号化する。

## 【2427】2. 4. 1 形式

リファレンスのエンコーディングは以下の形式の1つを取る。

## 【2428】一般化

”ジェネラルリファレンスエンコーディング (general reference encoding)” は、リファレンスの符号化の開始からオブジェクトの符号化の開始までオクテットで表

した”オフセット (offset)” によってオブジェクトを表す (プロテクトされるかまたはプロテクトされない) リファレンスを符号化する。オフセットは、オブジェクトの符号化がリファレンスの符号化の後になるか先になるかに従ってそれぞれ正または負となる。オフセットは負とし、正のオフセットは留保される。

## 【2429】Voided

”ボイデッドリファレンスエンコーディング (voided reference encoding)” は、ボイドされるリファレンスを符号化する。

## 【2430】Interchange

”インターチェンジリファレンスエンコーディング (interchange reference encoding)” は、オブジェクトのダイジェストと、オブジェクトが1つの例であるインスタンスへのサイティションによって、相互変換されたオブジェクトを表すリファレンスを符号化する。

## 【2431】Class

”クラスリファレンスエンコーディング (class reference encoding)” は、クラスが予め定義されていれば、クラスの数値コードによってまたクラスがユーザによって定義されていればクラスへのサイティションによって、クラスを表すリファレンスを符号化する。

## 【2432】Procedure

”プロシージャリファレンスエンコーディング (procedure reference encoding)” は、ユーザによって定義されたクラスのインプリメンテーションにあるプロシージャへのリファレンスを符号化する。プロシージャリファレンスの符号化は予め定義されたフレームFの”プロシージャ (procedure)” 属性のみをエンコードするために用いられるべきである。プロシージャリファレンスの符号化は、”F” が1つのアイテムである”フレーム (frames)” の”F” の直下において、プロシージャフレーム”B” の”プロシージャ (procedure)” の属性中のプロシージャの位置を、整数によって表現する。この整数は、整数が1に等しいかこれより大きい場合に、位置自身でありまたは、他の場合には、非減数である位置と減数であるFの属性”位置 (position)” との算術差に1を足した値に等しい。

## 【2433】2. 4. 2 規則

あるオブジェクトへのリファレンスのリファレンス符号化の形式は、次のルールに従って選択されるものとし、これらの規則は次に示す順序で適用されるものとする。

【2434】1. ボイドされたリファレンスの符号化はリファレンスがボイドにされた場合選択されねばならない。

【2435】2. オブジェクトが予め定義されたクラスである場合、クラスリファレンス符号化を選択しなければならない。

【2436】1. リファレンスがボイドされた場合、ボイドされたリファレンスエンコーディングを選択する。

【2437】2. オブジェクトは予め定義されたクラスである場合、クラスリファレンス符号化を選択する。

【2438】3. オブジェクトがテレパーセルの目的点に存在していると推定されるのではなく、テレパーセルの一部分として符号化されている場合、一般的なりファレンス符号化を選択する。

【2439】4. オブジェクトがユーザによって定義されたクラスであるかまたはユーザによって定義されたクラスインプリメンテーションのプロシージャである場合それぞれクラスリファレンス符号化かまたはプロシージャリファレンス符号化を選択する。

【2440】5. (その他の場合は) 相互変換リファレンス符号化を選択する。

【2441】注: オブジェクトが、テレパーセルの目的点に存在していると推定されるのではなく、テレパーセルの一部分として符号化されている場合にのみ、規則3が適用される。

【2442】3 エンコーディングスペシフィケーション (ENCODING SPECIFICATIONS)

テレパーセルはこのアペンディックスのこのセクションにおいて定義される。サブセクションは、テレパーセル、オブジェクト、属性及びリファレンス符号化に向けられる。

【2443】3. 1 コンバージョン

テレパーセルはその各々がゼロまたはそれ以上のオクテットである一連のトークンである。テレパーセルはトークンを連鎖化することによって得られたオクテットストリングである。

【2444】テレパーセルは、下記のシンタクティックルール及び付随するセマンティックルールに従う。BNF (Backus-Naur又はBackus normal form) に与えられた規則は、任意のトークンを括弧 ( " [ " 及び " ] " ) によって囲む。

【2445】表B. 1は、オブジェクト及びリファレンス符号化の形式にコードを割り当てる。これらのコードは下記のセクションにおいてタグとして用いられる。

【2446】

【表28】

表 B. 1	
- 5 0	プロシージャリファレンス (ProcedureReference)
- 5 1	インターチェンジリファレンス (InterchangeReference)
- 5 2	アンプロテクティドリファレンス (UnprotectedReference)
- 5 3	プレデファインドリファレンス (PredefinedReference)
- 5 4	プロテクティドリファレンス (ProtectedReference)
- 5 5	プロテクティドオブジェクト (ProtectedObject)
- 5 6	-
- 5 7	-
- 5 8	ユーザ定義クラスリファレンス (UserDefinedClassReference)
- 5 9	ビデオリファレンス (VideoReference)
- 6 0	マックスインス (Maxins)

注: 下記のセクションは、この開示のアペンディックスAのセクション5. 1及び5. 3のコンベンションに従う。

【2447】3. 2 テレパーセル (Teleparcel)  
テレパーセルは次の規則に従う。

【2448】Teleparcel ::= unsignedNumber unsignedNumber Object

第1及び第2の " アンサインドナンバ (unsignedNumber) " は、テレパーセルがそれに準拠する符号化規則のメジャーバージョン及びマイナーバージョンの数をそれぞれ符号化する。 " オブジェクト (Object) " は、テレパーセルのサブジェクト、サブジェクトのコンポーネントとを符号化する。

【2449】注: このアペンディックスが規定する符号化規則のメジャーバージョン及びマイナーバージョン

の数はそれぞれ0及び5である。

【2450】3. 3 オブジェクトエンコーディング (Object Encoding)

オブジェクト符号化は定義された形式が反映する次の規則に従う。

【2451】Object ::= ExecutedObject | PredefinedObject

| UserDefinedObject | ProtectedObject

3. 3. 1 エグゼキューティドオブジェクト (Executed Object)

実行されるオブジェクトの符号化は、この開示のアペンディックスAのセクション5. 1及び5. 3に述べた規則に従う。

【2452】 " エグゼキューティッドオブジェクト (ExecutedObject) " の第1トークンであるタグは、実行さ

れたオブジェクトを特定し、クラスのコードを2回符号化することによってオブジェクトのクラスを明らかにする。

【2453】注：実行されるオブジェクトは、2進テレスクリプトと同様に符号化される。

【2454】3. 3. 2 プレデファインドオブジェクト (Predefined Object)

予め定義されたオブジェクトの符号化は次の規則に従う。

【2455】PredefinedObject ::= tag Palette  
タグがオブジェクトを予め定義されたものとして特定し、クラスのコードを2回符号化することによってオブジェクトのクラスを明らかにする。”パレット (Palette)”は、そのオブジェクトが1つの例であるクラスについて、1次符号化パレットに依存する。

【2456】3. 3. 3 ユーザデファインドオブジェクト (User-defined Object)

ユーザによって定義されたオブジェクトの符号化は次の規則に従う。

【2457】UserDefinedObject ::= [Palette Mixins]  
tag Item Item

Palette Mixins ::= tag

第1の”タグ (tag)”はクラスのコードの2倍に1を

ProtectedObject ::=  
ProtectedObjectItself ::=

”タグ (tag)”は、プロテクトされるリファレンスを特定する。”ExecutedObject”、”PredefinedObject”または”UserDefinedObject”はこのオブジェクトを符号化する。

【2462】3. 3. 5 パレット (Palette)  
符号化パレット符号化は次の規則に従う。

【2463】

Palette ::= Attributes [Mask Attributes]

Mask ::= unsignedNumber

Attributes ::= [Attribute Attributes]

第1の”パレット (palette)”は、パレットを定義する表にリストされている順序で、パレットが不可欠と宣言する0以上の属性を符号化する。表は、”アトリビュート (Attributes:以下、単にアトリビュートとだけ記す)”の発生回数を固定させる。

【2464】パレットはどれかの属性を任意であると宣言した場合にのみ、”Mask”及び第2の”アトリビュート”が存在する。第2の”アトリビュート”は、ゼロまたはそれ以上の任意の属性を符号化する。”Mask”は、”アトリビュート”の発生回数を固定する。

【2465】”Mask”は、どの任意の属性が第2の”アトリビュート”を符号化するかを示す整数を符号化する。パレットの表は、範囲 [1, n] 中の任意の属性の

足したコードを符号化することによって、オブジェクトが1つの例であるクラスを明らかにする。

【2458】第1の”アイテム (Item)”は、オブジェクトのクラスを符号化し第2の”アイテム (Item)”はそのクラスのユーザによって定義されたインプリメンテーションスーパークラスにその標準的な順序でアイテムを対応しているリストを符号化する。各々のアイテムそれ自身はリスト、すなわち、クラスインプリメンテーションの属性”プロパティ (properties)”の識別子の順序によるアイテムが対応するクラスに固有のインスタンスのプロパティのリストである。最後の1以上のプロパティは省略できその場合はゼロが意味される。

【2459】第1の”パレット (Palette)”は、2次符号化パレットに依存し、”ミキシン (Mixins)”が存在する場合にのみ存在する。第2の”パレット (Palette)”は、オブジェクトがその一例であるクラスについて1次パレットに依存する。

【2460】3. 3. 4 プロテクトドオブジェクト (Protected Object)

プロテクトされるオブジェクトの符号化は次の規則に従う。

【2461】

tag ProtectedObjectItself

ExecutedObject |

PredefinedObject |

UserDefinedObject

番号を定める。第2の”アトリビュート”は、整数の表示  $2^{i-1}$  を加重した値は1になる場合にのみ番号”i”が属性の符号化を含む。実際に符号化された属性は、”i”の増大する順序で、第2の”アトリビュート”中に表れる。”n”によって論理的に必要とされるが”Mask”から物理的に不在であるビットはゼロと見なされる。物理的に存在しているが論理的に必要とされないビットは無視される。

【2466】3. 4 アトリビュートエンコーディング (Attribute Encoding)

属性符号化は規定された形式を反映する次の規則に従う。

【2467】Attribute ::= Boolean | DialString | Integer | Item | List | OctetString | String

3. 4. 1 ブーリアン (Boolean)

ブール属性の符号化は次の規則に従う。

【2468】Boolean ::= empty

ブーリアンが”true”の時にのみブーリアン属性符号化が存在する。

【2469】注：符号化の存在は符号化を網羅するマスクによって示される。

【2470】注：従ってパレットのブーリアン属性は必然的に任意である。

### 【2471】3. 4. 2 ダイアルストリング (Dial String)

ダイアルストリング符号化は次の規則に従う。

【2472】DialString ::= OctetString

“OctetString”は、ダイヤルストリング“D”によって決定し、同じ長さであるニブルのリスト“L”によって決定する、オクテットストリング“O”をエンコードする。

【2473】“L”のニブルは、同じ位置にある“D”のキャラクタによって定められた範囲[0, 14]中の整数を符号なしで符号化する。数(“0”-“9”)は0-9を、ダッシュ(“-”)は13を定め、スペース(“ ”)は14を定める。位置2i-1または2iにある“L”のニブル(“i”は少なくとも1である)は、位置“i”の“O”のオクテットのそれぞれビット7-3または3-0を占める。ビット7または3はそれぞれMSBであり、ビット4または0はそれぞれLSBである。“D”の長さ従って“L”が奇数である場合にのみ、“O”の最後のオクテットのビット3-0は同じ仕方では15を符号化する。

【2474】注：従ってダイヤルストリングは、2進化十進数(BCD)の形で符号化される。

### 【2475】3. 4. 3 整数 (Integer)

整数属性符号化は次の規則に従う。

【2476】Integer ::= signedNumber

“サインドナンバ(signedNumber)”は整数を符号化する。

【2477】注：これは整数の2進テレスク립ト符号化であるが、例外として、符号化を整数の一般的な符号化として特定する最初のトークンは存在しない。

### 【2478】3. 4. 4 アイテム (Item)

アイテム属性符号化は次の規則に従う。

【2479】Item ::= Object | Reference

“オブジェクト(Object)”または“リファレンス(Reference)”は、それぞれオブジェクトまたはオブジェクトへのリファレンスを符号化する。

### 【2480】3. 4. 5 リスト (List)

リスト属性符号化は次の規則に従う。

【2481】List ::= unsignedNumber Items

Items ::= [Item Items]

“アンサインドナンバ(unsignedNumber)”及び“アイテム(Items)”は一緒にリストを符号化する。前者は後者の“アイテム(Item)”の発生回数を符号化し、後者は増大する位置の順序でリスト中のアイテムを符号化する。

GeneralReference ::=

UnprotectedReference ::=

ProtectedReference ::=

“タグ(tag)”は、一般的リファレンスをそのよう

### 【2482】3. 4. 6 オクテットストリング (Octet String)

オクテットストリング属性符号化は次の規則に従う。

【2483】OctetString ::= unsignedNumber Octets

“アンサインドナンバ(unsignedNumber)”及び“オクテット(Octets)”は、この開示のアペンディックスAのセクション5. 3. 2に規定されたオクテットストリングを符号化する。

【2484】注：これはオクテットストリングの2進テレスク립ト符号化であるが例外として、オクテットストリングの符号化を特定する最初のトークンが存在しない。

### 【2485】3. 4. 7 スタック (Stack)

スタック属性符号化はリスト属性符号化と同一の規則に従うが、例外として、スタックのアイテムは位置の減少する順序で符号化される。

### 【2486】3. 4. 8 ストリング (String)

ストリング属性符号化は次の規則に従う。

【2487】

String ::= unsignedNumber characters

“アンサインドナンバ(unsignedNumber)”及び“キャラクターズ(characters)”は、この開示のアペンディックスAのセクション5. 3. 2に規定されたストリングと一緒に符号化する。

【2488】注：これはストリングの2進テレスク립ト符号化であるが、例外として、符号化をストリングのものとして特定する最初のトークンが存在しない。

### 【2489】3. 5 リファレンスエンコーディング

(Reference Encoding)

リファレンス符号化は、規定された形式を反映する次の規則に従う。

【2490】

Reference ::= GeneralReference |  
voidedReference |  
InterchangeReference |  
ClassReference |  
ProcedureReference

### 3. 5. 1 ゼネラルリファレンス (General Reference)

一般的リファレンスの符号化は次の規則に従う。プロテクトされないリファレンスに対してはある符号化が定義され、プロテクトされるリファレンスについては別の符号化が定義される。

【2491】

UnprotectedReference |

ProtectedReference

tag signedNumber

tag signedNumber

なものとして特定し、またプロテクトされないものもし

くはプロテクトされるものとして特定する。”サインドナンバ (signedNumber)” は、リファレンスの符号化からリファレンスされるオブジェクト符号化までオフセットを符号化する。

【2492】3. 5. 2 ボイディドリファレンス (Voided Reference)

ボイドされたリファレンスの符号化は次の規則に従う。

【2493】VoidReference ::= tag

”タグ (tag)” は、ボイドされたリファレンスをそのようなものとして特定する。

【2494】3. 5. 3 インターチェンジリファレンス (Interchange Reference)

相互変換リファレンス符号化は次の規則に従う。

【2495】

ClassReference ::=

PredefinedClassReference ::=

UserDefinedClassReference ::=

”タグ (tag)” は、クラスリファレンスをそのようなものとして特定し、またクラスを予め定義されたものもしくはユーザによって定義されたものとして特定する。クラスが予め定義されていた場合、”アンサインドナンバ (unsignedNumber)” は、この開示のアペンデックスAのセクション5. 4. 1またはこのアペンディックスの下記の表B. 2によるクラスに割り当てられるコードに符号化される。

【2498】クラスがユーザによって定義されていれば、”アイテム (Item)” はクラスのサイテーションを符号化する。

【2499】3. 5. 5 プロシージャリファレンス (Procedure Reference)

プロシージャリファレンス符号化は次の規則に従う。

【2500】

ProcedureReference ::= tag signedNumber

”タグ (tag)” は、プロシージャのリファレンスをそのようなものとして特定する。”サインドナンバ (signedNumber)” は先に述べたようにプロシージャの位置を

InterchangeReference ::= tag Item Item

”タグ (tag)” は、相互変換リファレンスをそのようなものとして特定する。第1の”アイテム (Item)” は、相互変換されたクラスへのサイテーションを符号化する。第2の”アイテム (Item)” は、相互変換されたオブジェクトのダイジェストを符号化する。

【2496】3. 5. 4 クラスリファレンス (Class Reference)

クラスリファレンス符号化は次の規則に従う。予め定義されたクラスについてはある符号化が定義され、ユーザによって定義されるクラスに対しては符号化が定義される。

【2497】

PredefinedClassReference |

UserDefinedClassReference

tag unsignedNumber

tag Item

特定する。

【2501】4 エンコーディングクラスサイズと属性 (ENCODING CLASS AND ATTRIBUTES)

テレスクリプトの符号化クラス及び属性はこのアペンディックスのこのセクションに定義されている。あるサブセクションは第1にクラスの符号化にそして次に属性の符号化にそれぞれ向けられる。

【2502】4. 1 コンベンションズ (Conventions)

Encoding Classes

符号化クラスは1つのグループを形成する。このグループを規定する以下のセクションは、この開示のアペンディックスAのセクション3. 2のコンベンションに従う。

【2503】表B. 2は、符号化クラスへの識別子のコードを割り当てる。

【2504】

【表29】

表B. 2	
144	キャッチフレーム (CatchFrame)
145	コレクションストリーム (collectionStream)
146	—
147	プリ定義フレーム (PredefinedFrame)
148	ゴーフレーム (GoFrame)
149	リピートフレーム (RepeatFrame)
150	レストリクトフレーム (RestrictFrame)
151	センドフレーム (SendFrame)
152	ユースフレーム (UseFrame)
153	ユーザ定義フレーム (UserDefinedFrame)
171	ホワイルフレーム (WhileFrame)

注：クラス”フレーム (Frame)” 及び”プロシージャ

フレーム (ProcedureFrame)” は、抽象的であるためコ

ードを必要としない。クラス”ダイヤルストリング (Dial String)”はそのインスタンスがそれ自身の符号化を持つためコードを必要としない。

【2505】注：符号化クラスの唯一つの目的が符号化属性を規定することにあるので、このアペンディックスは、アペンディックスAがそのようなセクション例えばセクション4. 2を各々の言語クラスに向けているにもかかわらず、各々の符号化クラスに大きな部分を割いていない。

#### 【2506】Encoding Attributes

符号化属性はある言語及び符号化クラスに固有である。符号化属性を定義する以下のセクションは、そのような属性がそれに対して固有である各々の予め定義されたクラスに1つのサブセクションを割いている。これらのクラスはアルファベット順序と考えられる。符号化属性の定義は、この開示のアペンディックスAのセクション4. 1に定めたコンベンションに従う。

#### 【2507】注：識別子のコード.....

識別子のコードは、符号化属性に割り当てられる必要はない。

#### 【2508】4. 2 エンコーディンググループ (Encoding Group)

オブジェクト (参照される)

- ・ コレクション
- ・ ・ リスト (順序付けされる)
- ・ ・ ・ コンストレインドリスト (コンストレインド)
- ・ ・ ・ ・ ストリング (ケースド、実行される)
- ・ ・ ・ ・ ・ Dial String
- ・ Frame
- ・ ・ Procedure Frame
- ・ ・ ・ Predefined Frame
- ・ ・ ・ ・ Catch Frame
- ・ ・ ・ ・ Repeat Frame
- ・ ・ ・ ・ Restrict Frame
- ・ ・ ・ ・ Use Frame
- ・ ・ ・ ・ While Frame
- ・ ・ ・ User-defined Frame
- ・ ・ Go Frame
- ・ ・ ・ Send Frame
- ・ Stream
- ・ ・ Collection Stream

#### 4. 2. 1 キャッチフレーム (Catch Frame)

Attributes

exception

”キャッチフレーム (catch frame)”は、そのサブジェクトメソッドが”キャッチ (catch)”の方法であり、その”プロシージャ (procedure)”の属性はオペレーションのレスポンドである、予め規定されたフレームである。

【2509】キャッチフレームの固有の属性は、オペレ

ーションのアーギュメントであるクラスである (属性”エクセプション (exception)”)。

#### 【2510】4. 2. 2 コレクションストリーム (Collection Stream)

Attributes

position

source

”コレクションストリーム (collection stream)”

は、そのアイテムがコレクションのアイテムであるストリームである。

【2511】コレクションストリームの固有の属性は、ストリングのソース (属性”ソース (source)”) 及びストリングの現在の位置 (属性”ポジション (position)”) である。”ソース (source)”は、そのアイテムがコレクションストリームによって生ずるコレクションである。”カレントポジション (current position)”は、コレクションストリームのソースを符号化する属性”アイテム (items)”中の位置である。

【2512】コレクションストリームは、ストリームの現在の位置においてそのアイテムを最後に作り出し、現在の位置の前方の位置 (複数) においてなんらかのアイテムをすでに作り出し、そして現在の位置の後方の位置 (複数) においてさらに別のアイテムをこれから作り出す。

【2513】注：コレクションストリームは、”ストリーム (stream)”オペレーションがコレクションによって遂行された時のその”ストリーム (stream)”オペレーションの結果である。

#### 【2514】4. 2. 3 ダイヤルストリング (Dial String)

”ダイヤルストリング (dial string)”は、その各々のアイテムが数字 (“0”-“9”)、ハイフン (“-”) またはスペース (“ ”) であるストリングである。

【2515】注：ダイヤルストリングはテレナンバにおいて用いられる。

#### 【2516】4. 2. 4 フレーム (Frame)

”フレーム (frame)”は、予め定義されるかまたはユーザによって定義された方法、フレームの”サブジェクトメソッド (subject method)”の遂行の現在の状態を記録するオブジェクトである。

#### 【2517】4. 2. 5 ゴーフレーム (Go Frame)

”ゴーフレーム (go frame)”はトリップを行なうことを課するフレームである。方法は、オペレーション”go”または”send”のための予め定められた方法であり、前者のオペレーションは、このクラスのあるインスタンスの場合であり、後者のオペレーションは、このクラスのサブクラスのインスタンスの場合である。

#### 【2518】4. 2. 6 プレデファインドフレーム (Predefined Frame)



” プレディファインドフレーム (predefined frame) ” は、そのサブジェクトメソッドが予め定義されており、従ってそのプロシージャがその方法ではないプロシージャフレームである。このクラスのあるインスタンスのサブジェクトメソッドは、オペレーション ” do ” 、 ” either ” 、 ” if ” 、 ” loop ” または ” select ” のための方法である。

【2519】注：前記のオペレーションの内のどれが適用されるかは、その実行によってオペレーションがリクエストされた識別子を吟味することによって定められる。

【2520】4. 2. 7 プロシージャフレーム (Procedure Frame)

Attributes

position

procedure

” プロシージャフレーム (procedure frame) ” は、プロシージャの遂行を課するフレームである。

【2521】プロシージャフレームの固有の属性は、遂行されているプロシージャ (属性 ” プロシージャ (procedure) ” ) と、プロシージャの現在の位置 (属性 ” 位置 (position) ” ) である。プロシージャの ” カレントポジション (current position) ” は、エンジンの現在実行中のアイテムの現在の位置である。

【2522】注：プロシージャはサブジェクトプロシージャの属性 ” プロシージャ (procedure) ” であるか、その方法がインプリメントするフィーチャのレスポндаであるか、またはフィーチャのアーギュメントである。

【2523】4. 2. 8 リピートフレーム (Repeat Frame)

Attributes

repetitions

repetitionsSoFar

” リピートフレーム (repeate frame) ” は、そのサブジェクトメソッドがオペレーション ” リピート (repeat) ” の方法であり、その属性 ” プロシージャ (procedure) ” がそのオペレーションのレスポндаである、予め規定されたフレームである。

【2524】リピートフレームの固有の属性は、開始された現在の遂行 (属性 ” レピティションズソーファ (repetitionsSoFar) ” ) 、及び最初にリクエストされた数 (属性 ” レピティションズ (repetitions) ” ) を含むフレームの属性 ” プロシージャ (procedure) ” の遂行の数である。

【2525】4. 2. 9 レストリクトフレーム (Restrict Frame)

Attributes

permit

” レストリクトフレーム (restrict frame) ” は、そのサブジェクトメソッドがオペレーション ” restrict ” の

ための方法であり、その属性 ” プロシージャ (procedure) ” がオペレーションのアーギュメントである予め定義されたフレームである。

【2526】レストリクトフレームの固有の属性は、サブジェクトメソッドのエンジンによる遂行の開始時にサブジェクトが持ったパーミット (属性 ” permit ” ) である。

【2527】4. 2. 10 センドフレーム (Second Frame)

Attributes

tickets

” センドフレーム ” は、そのサブジェクトメソッドがオペレーション ” send ” のための予め定義された方法であるゴーフフレームである。

【2528】センドフレームの固有の属性は、最初にオペレーションのアーギュメントにあった1またはそれ以上のチケット (属性 ” tickets ” ) を含む。

【2529】4. 2. 11 ユースフレーム (Use Frame)

Attributes

resource

” ユーズフレーム (use frame) ” は、そのサブジェクトメソッドがオペレーション ” ユース (use) ” の方法であり、その属性 ” プロシージャ (procedure) ” がオペレーションのアーギュメントの1つである予め定義されたフレームである。

【2530】ユーズフレームの固有の属性は、オペレーションのレスポнда (属性 ” resource ” ) である。

【2531】4. 2. 12 ユースデファインドフレーム (User-defined frame) Attributes

responder

stack

variables

” ユーザディファインドフレーム (user-defined frame) ” は、そのプロシージャがフレームのサブジェクトメソッドのプロシージャであるプロシージャフレームである。サブジェクトメソッドはユーザによって定義され、必然的に、インプリメンテーションの属性 ” インスタンスメソッズ (instanceMethods) ” または ” セットメソッズ (setMethods) ” の値である。ユーザによって定義されたフレームの固有の属性は、その方法のレスポнда (属性 ” responder ” ) 、スタック (属性 ” stack ” ) 、及び変数 (属性 ” variables ” ) であり、後者は、方法の属性 ” variables ” における識別子の順序である。

【2532】レスポндаは、ユーザによって定義されたフレームがその属性 ” frames ” の1つのアイテムであるエージェントである。

【2533】注： ” I ” (もしあれば) が、エージェントの属性 ” frames ” 中のユーザによって定義されたフレ

ームの下方のフレーム” B” (もしあれば) の属性” プロシージャ (procedure)” 中の1つのアイテムであるとする。特に、” I” が、属性” プロシージャ” 中のその位置が” B” の属性” 位置” であるアイテムであるとする。” I” は (a) 存在せず、サブジェクトメソッドはオペレーション” live” のためのものであり、(b) モディファイヤ” セットアトリビュート (setAttribute)” によって先行される1つの識別子であり、サブジェクトメソッドはインプリメンテーションの属性” セットメソッズ (setMethods)” 中の1つの値であり (c) このように先行されない1つの識別子であり、サブジェクトメソッドは、インプリメンテーションの属性” インスタンスメソッズ (instanceMethods)” 中の1つの値であり、または (d) セレクタ” エスカレート (escalate)” であり、サブジェクトメソッドは上記 (b) または (c) に示した通りとする。

【2534】4. 2. 13 ホワイルフレーム (While Frame)

Attributes

isPrecondition

precondition

” ホワイルフレーム (while frame)” は、そのサブジェクトメソッドがオペレーション” while” のための方法であり、その属性” プロシージャ (procedure)” がそのオペレーションのレスポンドである予め定義されたフレームである。

【2535】ホワイルフレームの固有の属性は、オペレーションのアーギュメント (属性” プレコンディション (precondition)” ) である実行されるオブジェクトと、レスポンドでなくアーギュメントが遂行されているか否か (属性” イズプレコンディション (isPrecondition)” ) を明らかにする。ホワイルフレームの属性” 位置 (position)” は、2つのプロシージャの内のどちらが遂行されているかに所属する。

【2536】4. 3 エンコーディングアトリビュート (Encoding Attribute)

符号化属性は次のように定義される。

【2537】4. 3. 1 エージェント (Agent)

次の符号化属性がこのクラスに固有である。

【2538】frames

sealed Stack[Frame]!;

レスポンドの継続中のフレーム。底部のフレームのサブジェクトメソッドは、オペレーション” live” のためのユーザによって定義された方法である。トップフレームのサブジェクトメソッドは、オペレーション” go” または” send” のための作り付けられた方法である。ゼロまたはそれ以上の他のフレームがある。

【2539】nativePermit:

sealed copied Permit[Nil];

レスポンドの固有のパーミットのレスポンドの保持する

パーミットと相似していれば、レスポンドの固有のパーミットであり、その他の場合にはゼロである。

【2540】4. 3. 2 キャッチフレーム (catch frame)

次の符号化属性はこのクラスに固有である。

【2541】exception:

sealed Class;

レスポンドのサブジェクトメソッドを遂行しているクラス (クラス” エクセプションException” またはそのサブクラス)。

【2542】4. 3. 3 コレクション (Collection)

次の符号化属性はこのクラスに固有である。

【2543】items:

sealed List[Object]!;

レスポンドのアイテム。特別のどのアイテムの位置も定義されていない。

【2544】4. 3. 4 コレクションストリーム (Collection Stream)

次の符号化属性はこのクラスに固有である。

【2545】position:

sealed Integer|Nil;

レスポンドがアイテムが生産していないかそのアイテムの全てを生産していない場合にはレスポンド中の現在の位置であり、さもなければゼロである。

【2546】source:

sealed Collection[Object]|Nil;

レスポンドがそのアイテムの全てを生産していなければ、レスポンドのソースでありさもなければゼロである。

【2547】4. 3. 5 ディクショナリ (Dictionary)

次の符号化属性はこのクラスに固有である。

【2548】associations:

sealed List[Object]!;

レスポンドのアイテムを形成するキー及び値。リストの位置  $2i-1$  及び  $2i$  にあるアイテムは、それぞれキーとその組み合わせられた値である。特別のどれかのキー値対の位置は規定されていない。

【2549】4. 3. 6 リスト (List)

次の符号化属性はこのクラスに固有である。

【2550】items:

sealed List[Object]!;

レスポンドのアイテム。

【2551】4. 3. 7 パターン (pattern)

次の符号化属性はこのクラスに固有である。

【2552】text:

シールされコピーされたストリング。

【2553】レスポンドのテキスト。

【2554】4. 3. 8 プロシージャフレーム (Procedure Frame)

次の符号化属性はこのクラスに固有である。

【2555】sealed Integer;

レスポンドの属性”プロシージャ (procedure)”中の現在の位置。

【2556】procedure:

sealed Procedure;

その実行をレスポンドが課するプロシージャ。

【2557】4. 3. 9 ランダムストリーム (Random Stream)

次の符号化属性はこのクラスに固有である。

【2558】seed:

sealed Integer;

レスポンドがいかなる論理も形成していない場合、レスポンドのシードであり、さもなければレスポンドが最後に生産したアイテムである。

【2559】4. 3. 10 リピートフレーム (Repeat Frame)

次の符号化属性はこのクラスに固有である。

【2560】repetitions:

sealed Integer;

オペレーション”repeat”のアーギュメントを介した、レスポンドのサブジェクトメソッドについて要求されたレスポンドの属性”プロシージャ (procedure)”の遂行の回数。

【2561】repetitionsSoFar:

sealed Integer;

これまでに開始された、現在の遂行を含めてレスポンドの属性”プロシージャ (procedure)”が遂行された回数。

【2562】4. 3. 11 レストリクトフレーム (Restrict Frame)

次の符号化属性はこのクラスに固有である。

【2563】permit:

sealed copied Permit;

レスポンドのサブジェクトメソッドの遂行の開始時ににおいてサブジェクトが保持したパーミット。

【2564】4. 3. 12 センドフレーム (Send Frame)

次の符号化属性はこのクラスに固有である。

【2565】tickets:

sealed List [Ticket]!;

レスポンドのサブジェクトメソッドのアーギュメントとして供給されたリスト中の1以上のチケット。どの特別のチケットの位置も規定されていない。

【2566】4. 3. 13 スタック (Stack)

次の符号化属性はこのクラスに固有である。

【2567】items:

sealed Stack [Object]!;

レスポンドのアイテム。

【2568】4. 3. 14 タイム (Time)

次の符号化属性はこのクラスに固有である。

【2569】dst:

sealed Integer;

クラス”タイム (time)”の”ニュー (new)”をリクエストし、その結果をカレンダー時間に変換することによる、生成されえたカレンダー時間の属性”dst”、いつどこでレスポンドが生成されたか。

【2570】second:

sealed Integer;

レスポンドがするのと同じ絶対時間点、”T”を表すが、その属性”セカンド (second)”のみがゼロにセットされた場合に時間”T<sub>0</sub>”を表すべき正常化されていないカレンダー時間の属性”セカンド (second)”。

【2571】注: T<sub>0</sub> は、全ての時間がそれに対して計られる時間である。T<sub>0</sub> は、グレゴリー暦記の分の開始に合致する。従ってこの属性は、TをT<sub>0</sub> から隔てる秒数である。この属性は厳密には時間がT<sub>0</sub> の後にあるか前にあるかによってそれぞれ正または負になる。

【2572】zone:

sealed Integer;

クラス”タイム (time)”の”ニュー (new)”をリクエストしその結果をカレンダー時間に変換することによって、生成されえたカレンダー時間の属性”zone”、いつどこでレスポンドが生成されたか。

【2573】4. 3. 15 ユースフレーム (Use Frame)

次の符号化属性はこのクラスに固有である。

【2574】resouce:

sealed Resource;

レスポンドのサブジェクトメソッドを遂行するリソース。

【2575】4. 3. 16 ユースデファインドフレーム (User-defined Frame)

次の符号化属性はこのクラスに固有である。

【2576】responder:

sealed Object;

レスポンドのサブジェクトメソッドのレスポンド。

【2577】stack:

sealed Stack [Object]!;

レスポンドのサブジェクトメソッドのスタック。

【2578】variables:

sealed List [Object]!;

レスポンドのサブジェクトメソッドの変数。

【2579】4. 3. 17 ホワイルフレーム (While Frame)

次の符号化属性はこのクラスに固有である。

【2580】isPrecondition:

sealed Boolean;

レスポンドの属性”プロシージャ”でなくその属性”プリコンディション (precondition)”が遂行されている

か否か。

【2581】 precondition:

sealed Executed;

オペレーション” repeat” のアーギュメント。

【2582】 ENCODING PALLETES

テレスク립トの符号化パレットはアペンディックスのこのセクションに定義されている。あるセクションは言語のクラスにまた別のセクションは符号化クラスにそれぞれ割かれている。あるネットワークのエンジンは次のことについて一致しなければならない。(i) 割り当てられたテレアドレスの属性” プロバイダ (provider)” が選択された方法。(ii) 割り当てられたテレネームの属性” オーソリティ (authority)” が選択される方法。これらの方法の各々の1つの実施例は、この開示のマイクロフィルムアペンディックスGに含まれたコンピュータソフトウェアにインプリメントされている。

【2583】 5. 1 コンベンションズ (Conventions)

ノンエンプティ (空でない) の1次パレットを用いてその例が符号化された各々の予め定義されたフレーバーに、1つのサブセクションが以下に向けられる。第1に言語フレーバーが次に符号化フレーバーがアルファベット順序で検討される。別のミックスインと題された余分のサブセクションは、その1つの二次パレットに向けられる。

【2584】 ほとんど各々のパレットは、その物が以下のようにラベルされ、その行が予め定められた属性を次のように定義する、表を用いて定義される。

【2585】 ・ #. 属性が命令的であれば” M” であり、属性が任意であれば属性に割り当てられた数字。

【2586】 ・ 属性。属性の識別子

・ センダー。アペンディックスAのセクションA.

2. 4のノーテーションを用いて定義されたアトリビュートの形式。その形式が” ブーリアン (Boolean)” であり、その2つのインスタンスの内の1つのみがパーミットされれば、” ブーリアン (Boolean)” の代わりに” 真 (true)” または” 偽 (false)” で表れる。

【2587】 ・ レシーバ。送出しているエンジンが符号化から属性を除かない場合はダッシュ (” -”) であり、送っているエンジンが属性を除かなければ、受けるエンジンが要求するべき形式のインスタンスである。

【2588】 5. 2 ランゲージクラスes (Language Classes)

言語クラスのための符号化パレットは次のように定義される。

【2589】 注: ある言語クラスがこのセクションにないのは次のように説明される。あるオブジェクトの属性クラス及びイズプロテクティッドはその符号化には表れるが属性としては表れない。オブジェクトはそのsize属性を計算する。あるプレースは移動できず従って符号化されない。あるプロセスのクラスはそのプロセスがエージェントまたはクラスでない限りユーザによって定義されない。あるストリングはその属性” カレント (current)”、” イズダン (isDone)” 及び” ネクスト (next)” を計算する。

【2590】 5. 2. 1 エージェント (Agent)

このクラスの例は表B. 3の1次パレットを用いて符号化される。

【2591】

【表30】

表 B. 3			
#	属性	送信側	受信側
M	name	Telename	-
M	permit	Permit	-
M	frames	Stack[Frame]!	-
1	brand	Object'	Assigned'
2	privateClasses	Set[Cited]	Cleared
3	nativePermit	Permit	Nil
4	priority	Integer	Permit <sup>2</sup>
5	contacts	Set[Contact]	Absent

<sup>1</sup> この属性は、エージェントがある領域内において移送されるならば、送り側のエンジンによって供給され、さもなければ受け側のエンジンによって供給される。

【2592】 <sup>2</sup> エージェントの属性” パーミット (permit)” の属性” プライオリティ (priority)”。

【2593】 5. 2. 2 アソシエーション (Association)

このクラスの例は表B. 4に示した1次パレットを用いて符号化される。

【2594】

【表31】

表 B. 4			
#	属 性	送信側	受信側
M	"key"	Object	-
M	"value"	Object	-

## 5. 2. 3 アトリビュート (Attribute)

【2595】

このクラスの例は表B. 5に示した1次パレットを用いて符号化される。

【表32】

表 B. 5			
#	属 性	送信側	受信側
1	"exception"	Set[Identifier!]	Cleared
2	"isPublic"	真	偽
3	"constraint"	Constraint	Constraintを参照
4	"isSet"	真	偽

1 あるクラスにおいてこれは"セット (set) [クラス (class)]"である。各々のクラスはクラス"エクセプション (Exception)" またはそのサブクラスである。

ime)

このクラスの例は表B. 6に示した1次パレットを用いて符号化される。

【2597】

## 【2596】5. 2. 4 カレンダータイム (Calendar T

【表33】

表 B. 6			
#	属 性	送信側	受信側
1	"day"	整数 (Integer)	なし (Nil)
2	"dayOfWeek"	整数 (Integer)	なし (Nil)
3	"dayOfYear"	整数 (Integer)	なし (Nil)
4	"dst"	整数 (Integer)	なし (Nil)
5	"hour"	整数 (Integer)	なし (Nil)
6	"minute"	整数 (Integer)	なし (Nil)
7	"month"	整数 (Integer)	なし (Nil)
8	"second"	整数 (Integer)	なし (Nil)
9	"year"	整数 (Integer)	なし (Nil)
10	"zone"	整数 (Integer)	なし (Nil)

## 5. 2. 5 サイトेशन (Citation)

【2598】

このクラスの一例は表B. 7に示した1次パレットを用いて符号化される。

【表34】

表 B. 7			
#	属 性	送信側	受信側
M	"title"	Identifier!	-
1	"author"	名前 (Telename)	なし (Nil)
2	"majorEditon"	整数 (Integer)	なし (Nil)
3	"minorEditon"	整数 (Integer)	なし (Nil)

## 5. 2. 6 クラス (Class)

【2599】

このクラスの例は表B. 8に示した1次パレットを用いて符号化される。

【表35】

表 B. 8			
#	属 性	送信側	受信側
M	"citation"	Interface	-
M	"interface"	Identifier!	-
1	"implementation"	Implementation	なし (Nil)

5. 2. 7 クラスデフィニション (Class Definition) 用いて符号化される。

n) 【2600】

このクラスの一例は表B. 9に示した1次パレットを用 【表36】

表 B. 9			
#	属 性	送信側	受信側
M	"interface"	Identifier!	-
M	"title"	Identifier!	-
1	"implementation"	Implementation	なし (Nil)
2	"majorEditon"	整数 (Integer)	なし (Nil)
3	"minorEditon"	整数 (Integer)	なし (Nil)

5. 2. 8 コレクション (Collection) 【2601】

このクラスの例は表B. 10に示した1次パレットを用 【表37】

用いて符号化される。

表 B. 10			
#	属 性	送信側	受信側
1	"item"	List[Object]!	Cleared

5. 2. 9 コンストレインディクショナリ (Constrained Dictionary) 用いて符号化される。

ained Dictionary) 【2602】

このクラスの例は表B. 11に示した1次パレットを用 【表38】

表 B. 11			
#	属 性	送信側	受信側
M	"constraint"	Constraint	-
M	"associations"	List[Object]!	-

5. 2. 10 コンストレインドリスト (Constrained List) 用いて符号化される。

List) 【2603】

このクラスの一例は表B. 12に示した1次パレットを 【表39】

表 B. 12			
#	属 性	送信側	受信側
M	"constraint"	Constraint	-
M	"items"	List[Object]	-

5. 2. 11 コンストレインドセット (Constrained Set) 用いて符号化される。

Set) 【2604】

このクラスの一例は表B. 13に示した1次パレットを 【表40】

表 B. 13			
#	属 性	送信側	受信側
M	"constraint"	Constraint	-
M	"items"	List[Object]	-

5. 2. 12 コンストレイント (Constraint) 用いて符号化される。

このクラスの一例は表B. 14に示した1次パレットを 【2605】

【表 4 1】

表 B. 1 4			
#	属 性	送信側	受信側
1	"isInstance"	真	偽
2	"isOptional"	真	偽
3 <sup>1</sup>	"classId"	Identifier!	'Object
3 <sup>1</sup>	"ofClass"	Class	'Object
4	"passage"	Identifier!	'byRef

1 属性"クラスアイディ (classId)"及び"オブクラス (ofClass)"は、内在的及び外在的にそれぞれ存在し、前者は後者の属性"サイテーション (citation)"の属性"タイトル (title)"である。定義クラス及び他の場所において、属性"クラスアイディ (classId)"及び"オブクラス (ofClass)"はそれぞれ外在的及び内在的に存在し後者はゼロである。クラスまたはクラス定義において、コンストレイントは、属性定義の属性"コンストレイント (constraint)"でありまたは、定義オペレーションの属性"アーギュメント (argument s)"の1つのアイテムである。これらのどの場合にお

いてもそのフィーチャの定義は、インタフェースの属性"クラスフィーチャーズ (classFeatures)"または"インスタンスフィーチャーズ (instanceFeatures)"中の値である。最後に、そのインタフェースは、クラスまたはクラス定義の属性"インタフェース (interface)"である。

【2606】5. 2. 13 コンタクト (Contact)  
このクラスの一例は表B. 15に示した1次パレットを用いて符号化される。

【2607】

【表 4 2】

表 B. 1 5			
#	属 性	送信側	受信側
1	"subject"	Process	なし (Nil)
2	"subjectClass"	Citation	なし (Nil)
3	"subjectName"	Telename	なし (Nil)
4	"subjectNotes"	Object	なし (Nil)

5. 2. 14 ディクショナリ (Dictionary)

【2608】

このクラスの例は表B. 16に示した1次パレットを用いて符号化される。

【表 4 3】

表 B. 1 6			
#	属 性	送信側	受信側
M	"associations"	List[Object]!	-

5. 2. 15 インプリメンテーション (Implementation)

用いて符号化される。

【2609】

このクラスの17例は表B. 17に示した1次パレットを

【表 4 4】

表 B. 1 7			
#	属 性	送信側	受信側
1	"classMethods"	Lexicon[Method]	クリアされる
2	"fromMethods"	Lexicon[Method]'	クリアされる
3	"instanceMethods"	Lexicon[Method]	クリアされる
4	"properties"	List[Identifier]!	クリアされる
5	"setMethods"	Lexicon[Method]	クリアされる
6	"superClasses"	List[Identifier]!	なし (Nil)
7	"toMethods"	Lexicon[Method]'	クリアされる
8	"vocabulary"	Lexicon[Citation]	なし (Nil)

1 あるクラスにおいてこれは"Dictionary [Class, Method]"である。

【2610】2 あるクラスにおいてこれは"List [Class]"である。

【2611】5. 2. 16 インターフェース (Interface)

用いて符号化される。

【2612】

このクラスの一例は表B. 18に示した1次パレットを

【表45】

表 B. 18			
#	属 性	送信側	受信側
1	"classFeatures"	Lexicon[Feature]	クリアされる
2	"instanceFeatures"	Lexicon[Feature]	クリアされる
3	"isAbstract"	真	偽
4	"sealedClassFeatures"	Set[Identifier!]	クリアされる
5	"sealedinstanceFeatures"	Set[Identifier!]	クリアされる
6	"superClasses"	List[Identifier!]¹	'Object
7	"vocabulary"	Lexicon[Citation]	クリアされる

¹ レシーバは、ある識別子のリスト、クラス"オブジェクト (Object)"のもの、を供給する。あるクラスにおいてこれは"List [Class]"であり、レシーバは、1つのクラスすなわちクラス"オブジェクト (Object)"のリストを供給する。

【2613】5. 2. 17 レキシコン (Lexicon)  
このクラスの一例は、クラス"ディクショナリ (Dictionary)"を符号化した1次パレットを用いて符号化される。

【2614】注：レキシコンの属性"コンストレイント (constraint)"は命令セットによって固定されている。

【2615】5. 2. 18 リスト (List)  
このクラスの一例は表B. 19に示した1次パレットを用いて符号化される。

【2616】

【表46】

表 B. 19			
#	属 性	送信側	受信側
M	"items"	List[Object]!	-

5. 2. 19 メソッド (Method)

【2617】

このクラスの一例は表B. 20に示した1次パレットを用いて符号化される。

【表47】

表 B. 20			
#	属 性	送信側	受信側
1	"procedure"	Procedure	クリアされる
2	"variables"	List[Identifier!]	クリアされる

5. 2. 20 ミックスイン (Mix-in)

【2618】

2次パレットは表B. 21のものである。

【表48】

表 B. 21			
#	属 性	送信側	受信側
1	"citation"	Citation	欠如
2	"constraint"	Constraint	欠如
3	name	Telename	欠如

注：ハッシュドオブジェクトの属性"ハッシュ (hash)"及び相互変換されたオブジェクトの属性"ダイジェスト (digest)"はオブジェクトによって計算される。

【2619】5. 2. 21 オペレーション (Operation)

このクラスの一例は表B. 22に示した1次パレットを用いて符号化される。

【2620】

【表49】



表 B. 2 2			
#	属 性	送信側	受信側
1	exceptions	Set[Identifier!]'	クリアされる
2	isPublic	真	偽
3	arguments	List[Constraint]	なし (Nil)
4	result	Constraint	なし (Nil)

1 あるクラスにおいてこれは、" Set [Class] " である。各々のクラスはエクセプションであるかまたはそのサブクラスである。

このクラスの一例は表 B. 2 3 に示した 1 次パレットを用いて符号化される。

【2 6 2 2】

【2 6 2 1】5. 2. 2 2 パッケージ (Package)

【表 5 0】

表 B. 2 3			
#	属 性	送信側	受信側
M	"citation"	Citation-	
M	"items"	List[Class]-	

注：パッケージの属性" コンストレイント (constrain  
1) " は命令セットによって固定されている。

用いて符号化される。

【2 6 2 4】

【2 6 2 3】5. 2. 2 3 パターン (Ptern)

【表 5 1】

このクラスの一例は表 B. 2 4 に示した 1 次パレットを

表 B. 2 4			
#	属 性	送信側	受信側
M	"text"	String	-

5. 2. 2 4 パーミット (Permit)

【2 6 2 5】

このクラスの一例は次表に示した 1 次パレットを用いて

【表 5 2】

符号化される。

表 B. 2 5			
#	属 性	送信側	受信側
1	allowance	整数 (Integer)	Of subject
2	canCharge	真	偽
3	canGo	真	偽
4	canProcreate	真	偽
5	canRestart	真	偽
6	canSend	真	偽
7	canTerminate	真	偽
8	deadline	Time	Of Subject
9	priority	整数 (Integer)	Of Subject

5. 2. 2 5 ペティション (Petition)

【2 6 2 6】

このクラスの一例は表中の 1 次パレットを用いて符号化  
される。

【表 5 3】

表 B. 2 6			
#	属 性	送信側	受信側
1	"agentClass"	サイティション (Citation)	なし (Nil)
2	"agentName"	テレネーム (Telename)	なし (Nil)
3	"maximumWait"	整数 (Integer)	なし (Nil)

5. 2. 2 6 ランダムストリーム (Random Stream)

【2 6 2 7】

このクラスの一例は表 B. 2 7 に示した 1 次パレットを  
用いて符号化される。

【表 5 4】

表 B. 27			
#	属 性	送信側	受信側
M	"seed"	整数 (Integer)	-

## 5. 2. 27 リソース (Resource)

【2628】

このクラスの一例は表B. 28に示した1次パレットを

【表55】

用いて符号化される。

表 B. 28			
#	属 性	送信側	受信側
1	"condition"	Identifier!	未定義 <sup>1</sup>
2	"conditions"	Set[Identifier!]	未定義 <sup>1</sup>

<sup>1</sup> 属性"コンディションズ (conditions)" は、属性"コンディション (condition)" に等しい1つの識別子を含む。この識別子は定義されていない。

【2629】注：サブジェクトがそのトリップを開始した時にリソースの使用を待っていたサブジェクト以外のどのプロセスも、後に残されてしまったので、もはやリソースを待っていない。

【2630】5. 2. 28 セット (Set)

このクラスの一例は、クラス"コレクション (Collection)" の一例がそれによって符号化された1次パレットを用いて符号化される。

【2631】5. 2. 29 スタック (Stack)

このクラスの一例は表B. 29に示した1次パレットを用いて符号化される。

【2632】

【表56】

表 B. 29			
#	属 性	送信側	受信側
M	"items"	Stack[Object]!	-

## 5. 2. 30 テレアドレス (Teleaddress)

【2633】

このクラスの一例は表B. 30に示した1次パレットを

【表57】

用いて符号化される。

表 B. 30			
#	属 性	送信側	受信側
M	"provider"	OctetString	-
1	"location"	String	なし (Nil)
2	"routingAdvice"	List[OctetString]	クリアされる

## 5. 2. 31 テレネーム (Telename)

【2634】

このクラスの一例は表B. 31に示した1次パレットを

【表58】

用いて符号化される。

表 B. 31			
#	属 性	送信側	受信側
M	"authority"	OctetString	-
1	"identity"	OctetString	なし (Nil)

## 5. 2. 32 テレナンバ (Telenumbar)

【2635】

このクラスの一例は表B. 32に示した1次パレットを

【表59】

用いて符号化される。

表 B. 32			
#	属 性	送信側	受信側
M	"country"	DialString	-
M	"telephone"	DialString	-
1	"extension"	DialString	なし (Nil)

## 5. 2. 33 チケット (Ticket)

[2636]

このクラスの一例は表B. 33に示した1次パレットを用いて符号化される。

[表60]

表 B. 33			
#	属 性	送信側	受信側
1	"travelNotes"	Object	なし (Nil)
2	"way"	Way	なし (Nil)
3	"desireWait"	整数 (Integer)	なし (Nil)
4	"destinationAddress"	Teleaddress	なし (Nil)
5	"destinationClass"	Citation	なし (Nil)
6	"destinationName"	Telename	なし (Nil)
7	"destinationPermit"	Permit	なし (Nil)
8	"maximumWait"	整数 (Integer)	なし (Nil)

## 5. 2. 34 チケットスタブ (Ticket Stub)

[2637]

このクラスの一例は表B. 34に示した1次パレットを用いて符号化される。

[表61]

表 B. 34			
#	属 性	送信側	受信側
1	"travelNotes"	Object	なし (Nil)
2	"way"	Way	なし (Nil)

## 5. 2. 35 タイム (Time)

[2638]

このクラスの一例は表B. 35に示した1次パレットを用いて符号化される。

[表62]

表 B. 35			
#	属 性	送信側	受信側
1	"dst"	Integer	0
2	"second"	Integer	0
3	"zone"	Integer	0

## 5. 2. 36 トリップエグセプション (Trip Exception)

用いて符号化される。

[2639]

このクラスの一例は表B. 36に示した1次パレットを

[表63]

表 B. 36			
#	属 性	送信側	受信側
M	"ticketStub"	TicketStub	なし (Nil)

## 5. 2. 37 アンエクスペクティドエクセプション (Unexpected Exception)

用いて符号化される。

[2640]

このクラスの一例は表B. 37に示した1次パレットを

[表64]

表 B. 37			
#	属 性	送信側	受信側
M	"exception"	Exception	-

## 5. 2. 38 ウェイ (Way)

[2641]

このクラスの一例は表B. 38に示した1次パレットを用いて符号化される。

[表65]

表 B. 3 8			
#	属 性	送信側	受信側
1	"authenticator"	Authenticator	なし (Nil)
2	"means"	Means	なし (Nil)
3	"name"	Telename	なし (Nil)

### 5. 3 エンコーディングクラス (Encoding Class)

es)

符号化クラスのための符号化パレットは以下に定義される。

このクラスの一例は表 B. 3 9 に示した 1 次パレットを用いて符号化される。

【2 6 4 3】

【2 6 4 2】 5. 3. 1 キャッチフレーム (Catch Frame)

【表 6 6】

表 B. 3 9			
#	属 性	送信側	受信側
M	"position"	整数 (Integer)	-
M	"procedure"	Procedure	-
M	"exception"	Class	-

### 5. 3. 2 コレクションストリーム (Collection Stream)

用いて符号化される。

【2 6 4 4】

このクラスの一例は表 B. 4 0 に示した 1 次パレットを

【表 6 7】

表 B. 4 0			
#	属 性	送信側	受信側
1	"position"	整数 (Integer)	なし (Nil)
2	"source"	Collection[Object]	なし (Nil)

### 5. 3. 3 プレデファインドフレーム (Predefined Frame)

用いて符号化される。

【2 6 4 5】

このクラスの一例は表 B. 4 1 に示した 1 次パレットを

【表 6 8】

表 B. 4 1			
#	属 性	送信側	受信側
M	"position"	整数 (Integer)	-
M	"procedure"	Procedure	-

### 5. 3. 4 リピートフレーム (Repeat Frame)

【2 6 4 6】

このクラスの一例は表 B. 4 2 に示した 1 次パレットを

【表 6 9】

用いて符号化される。

表 B. 4 2			
#	属 性	送信側	受信側
M	"position"	整数 (Integer)	-
M	"procedure"	Procedure	-
1	"repetitionsSoFar"	整数 (Integer)	1
2	"repetitions"	整数 (Integer)	1

### 5. 3. 5 レストリクトフレーム (Restrict Frame)

【0 0 0 0】

このクラスの一例は表 B. 4 3 に示した 1 次パレットを用いて符号化され

【2 6 4 8】

【2 6 4 7】る。

【表 7 0】



- 2. 8      トランスファ
- 2. 9      トランスファユニット
- 2. 10     テスティネーション
- 2. 11     トランスファグループ
- 2. 12     ウェイ
- 2. 13     ミーンズ
- 2. 14     リザーベーション
- 2. 15     ランク
- 3.        コミュニケーションクラシース
- 3. 1      リザーベーション
- 3. 2      リザーバブルミーンズ
- 3. 3      PSTNミーンズ
- 3. 4      エグジスティングコネクションミーンズ
- 4.        テレ스크リプトにおけるインターフェース
- 4. 1      エンジン
- 4. 2      プラットフォーム
- 4. 3      デスティネーション
- 4. 4      フェイラー
- 5.        ルーティング
- 6.        C++におけるインターフェース
- 7.        テレ스크リプトのインターフェースオブジ  
エクト
- 7. 1      インターフェースコンベンション
- 7. 2      Tsキャラクタ
- 7. 3      Tsデスティネーション
- 7. 4      Tsデスティネーションリスト
- 7. 5      Tsエグジスティングコネクションミーン  
ズ
- 7. 6      Tsインテジャ
- 7. 7      インテジャリスト
- 7. 8      Tsミーンズ
- 7. 9      Tsオブジェクトスペシファイヤ
- 7. 10     Tsオクテット
- 7. 11     Tsオクテツ
- 7. 12     TsPSTNミーンズ
- 7. 13     Tsリザーバブルミーンズ
- 7. 14     Tsリザーベーション
- 7. 15     Tsストリング
- 7. 16     Tsテレアドレス
- 7. 17     Tsテレネーム
- 7. 18     Tsテレナンバ
- 7. 19     Tsウェイ

## 1. 序

### 概観

このアベンディックスは、テレ스크リプトエンジンがその上で走るプラットフォームからテレ스크リプトエンジンが得ることができる“communication services”を定義する。2つのエンジンは、テレ스크リプトのオペレーション“go”及び“send”によって要求された、1つのエンジンの1つの場所から別のエンジンの別の場所へ“ag

ent”の表示を移送させるためにこれらのサービスを使用する。

【2658】このアベンディックスは、2つの主な部分からなる。第1に、このアベンディックスは、エンジンとプラットフォームとの間のインタフェースを記述するためにテレ스크リプトを用いて、テレ스크リプトエンジンが抽象的な仕方が必要とするコミュニケーションサービスを記述する。第2にこのアベンディックスがこれらのサービスを提供するC++“Application Programming Interface”(API)にドキュメンテーションを与える。

【2659】このアベンディックスは7つのセクションに分けられる。セクション1はこの序である。セクション2はインタフェースの基本的なコンセプトを記述する。セクション3は多少の追加的なテレ스크リプトクラスを記述する。セクション4はテレ스크リプトのインタフェースを記述する。セクション5はトランスファーを一層詳細に説明する。セクション6はインタフェースをインプリメントするC++APIを記述する。セクション7はAPIにおいて使用されるテレ스크リプトクラスのC++表示を記述する。

### 【2660】参考文献

このアベンディックスがこの開示のアベンディックスAに依存する。

### 【2661】2. インターフェースの概念

このアベンディックスのこのセクションは、テレ스크リプトコミュニケーションおよびコミュニケーションインタフェースがない基本的なコンセプトを記述する。ここに記述されるオブジェクトの多くは、テレ스크リプトの命令セットまたはエンジンのコミュニケーションインタフェース中に直接の表示を持つ。2. 1テレネーム(Telename)

テレ스크リプトとプレイスの間、特にエンジンの間のオブジェクトの運動を支持する。テレ스크リプトの世界の各々のプレイスには別々の“telename”が割り当てられている。“telename”は2つの属性すなわち“authority”と“identity”を持つ。

【2662】テレ스크リプトは割り当てられたものであってもそうでなくてもテレネームを支持する。テレ스크リプトエンジンによってオブジェクトに組み合わせられた“assigned telenames”は、次にオーソリティとアイデンティティの両方を含んでいる。“assigned”されていないテレネームは、1以上のテレ스크リプトオブジェクトを特定することを意図している。“assigned”されていないテレネームは、その属性アイデンティティを省略することができ、その場合にはテレネームは実際にいくつかのオブジェクトを特定する。

【2663】2. 2 ネットワーク(Network)

テレ스크リプト“Network”は1以上の領域を含む。ネットワークはその領域の全てを完全に相互接続できず、

これらの領域の任意の2つの間の"point-to-point"情報の伝達は可能ではないことがありうる。しかしネットワークが任意の2つの領域の間の情報の"store-and-forward"伝達を提供しうる程度に領域が相互接続される。

#### 【2664】2.3 エンジン (Engine)

"engine"は、テレクリプトの命令セットのアブストラクションをインプリメントするコンピュータプロセスである。エンジンは他のエンジンと通信することによって部分的にこれを行なう。エンジンは割り当てられたテレネームによって表される。

#### 【2665】2.4 プラットフォーム (Platform)

"platform"は、1以上のエンジンを支持するハードウェアおよびソフトウェアのコレクションである。プラットフォームは同一のプラットフォームおよびそのプラットフォームが接続された他のプラットフォーム上にあってエンジンが他のエンジンと通信することを可能にする。

#### 【2666】2.5 リージョン (Region)

"region"は個人または団体によって操作されるエンジンおよび支持プラットフォームを含む。

【2667】1つの領域は、その領域が含むプラットフォームを完全に相互接続する。従って領域は任意の2つのプラットフォームの間の情報をポイントからポイントに伝達することを可能にする。各々の領域は領域へのアクセスおよび領域から離去することを可能にする1以上のプラットフォームを含む。領域が相互接続されて1つのネットワークを形成するのはこれらのプラットフォームによる。ある領域は属性アイデンティティが省略されたテレネームによって表される。

【2668】2.6 アウトポスト (Outpost)  
ある領域は別の領域中にアウトポストを有しうる。"Outpost"は、ある領域からアウトポスト領域まで交信をルーティングする、その領域中のプレイスである。

#### 【2669】2.7 ドメイン (Domain)

"domain"は他の領域中のそのアウトポストの全てを含んだ1つの領域である。

#### 【2670】2.8 トランスファ (Transfer)

"transfer"は、1つのエンジンすなわち"source engine"から別のエンジンすなわち"destination engine"へのトランスファユニットへの伝送である。

【2671】トランスファユニットは、トランスファをルーティングするために用いることができる"destination"を特定する。さらに、トランスファについて使用されるルートおよび"transport"に影響することができるウェイを設けることができる。

#### 【2672】2.9 トランスファユニット (Transfer Unit)

"transfer Unit"はトランスファにおいてエンジンの間に伝達される。トランスファユニットはオクットストリングと目的点(デスティネーション)とから成る。これらはエンジンの間において端部間に伝送される唯一のデータアイテムである。

【2673】"originating platform"へのトランスファを特定し目的点においてのトランスファを受けることに、別のデータが関与する。

#### 【2674】2.10 デスティネーション (Destination)

"destination"はトランスファユニットの最終的な目的点を記述する。

#### 【2675】2.11 トランスファグループ (Transfer Group)

テレクリプト"send"オペレーション(アベンディックスAに記載される)は同一のオクテットストリングがいくつかの目的点に供与されることに結果する。"transfer group"は、トランスファユニットとそれに組み合わされた目的点と、伝送されているオクテットストリングの1つのコピーからなるコレクションである。

【2676】トランスファユニットは論理的にそのコンポーネントトランスファユニットを表している。プラットフォームは単一のトランスファグループを複数のより小さなトランスファグループおよび/または単一のトランスファユニットに分解することができる。

#### 【2677】2.12 ウェイ (Way)

"way"は、1つの領域またはエンジンと交信する手段と、それをするために用いられる"authenticating information"とを特定する。エンジンからのトランスファは、トランスファの通信要求を規定するウェイすなわち"way out"を特定することができる。到来するトランスファはエンジンにウェイすなわち"way back"を供与し、このウェイはオリジナルソースに向かう次のトランスファの戻り口として使用することができる。

#### 【2678】2.13 ミーンズ (Means)

"means"は、トランスファたとえば有線もしくは無線ネットワークに使用される搬送の形式を特定する。ミーンズは、"quality of service"パラメータまたは"PSN"コネクションのためのテレフォンナンバのような搬送に特異のデータを収容する。

【2679】"reservavle means"は、留保が尊重される搬送形式を特定する。

#### 【2680】2.14 リザベーション (Reservation)

"reservation"は、トランスファの結果としてプラットフォームによって設立されたコネクションがある期間の間保持されることの要求である。リザベーションは、多重にトランスファのためにあるコネクションが使用されることを許容するためのヒントである。

【2681】リザベーションは、トランスファが留保可

能な手段を有するウェイを特定する時にのみ意味を持つ。

#### 【2682】2. 15 ランク (Rank)

あるトランスファは”privileged”と考えることができる。”rank”はプリビレッジ(特典)の表示である。それは、整数であり、値0は特権のないことを意味する。0でない値の意味はソース領域特異である。領域間のトランスファにおいてランクは、変更または無視することができるヒントと見做すことができる。

#### 【2683】3. コミュニケーションクラス (Communication Classes)

このセクションは、この開示のアペンディックスAに存在していない別のテレスクリプトクラスを定義する。

#### 【2684】3. 1 リザベーション (Reservation)

オブジェクト

・Reservation

クラス

Reservation

インタフェース 0 = (···);

Public Instance Attributes

id;

インジヤ。

プラットフォームによって作り出された留保のための識別子

#### 3. 2 リザーバブルミーンズ (Reservable Means)

オブジェクト

・ミーンズ

・・Reservable Means

クラス

ReservableMeans:

アブストラクト・インタフェース 0 = (···);

Public Instance Attributes

reservation;

リザーベーション。

【2685】プラットフォームによって作り出されたリザベーションのための識別子。

#### 【2686】3. 3 PSTNミーンズ (PSTN Means)

オブジェクト

・ミーンズ

・・留保可能なミーンズ

・・・PSTN Means

クラス

PSTNMeans:

インタフェース (ReservableMeans) = (···);

Public Instance Attributes

telephoneNumber;

テレナンバ

このトリップのために使用されるテレホンナンバ。

#### 【2687】3. 4 エグジスティングコネクションミーンズ (Existing Connection Means)

オブジェクト

・ミーンズ

・・留保可能なミーンズ

・・・Existing Connection Means

クラス

ExistingConnectionMeans:

インタフェース (ReservableMeans) = (···);

Public Instance Attributes

connectionID;

コネクト。

あるコネクションを特定するためにプラットフォームが使用しうる識別子。

#### 【2688】4. テレスクリプトにおけるインターフェース (INTERFACE INTELESCRIPT)

このセクションではエンジンとそのプラットフォームとの間のインタフェースは、実際にはそうでないがあたかも両方ともテレスクリプトオブジェクトであるかのように特定される。この定義は族と考えられることが意図されている。すなわちこの定義は、エンジンおよびプラットフォームをインプリメントするために使用された特別のプログラミング言語とは無関係であることが意図されている。

【2689】以下のテレスクリプトクラスの定義はこのインタフェースにおいて用いられる。これらの定義についてはアペンディックスAおよびこのアペンディックスのセクション3参照。

#### 【2690】Integer

List

OctetString

Telename

Teleaddress

Way

Reservation

#### 4. 1 エンジン (Engine)

オブジェクト

・Engine

クラス

Engine:

インタフェース 0 = (···);

Public Instance Operations

transferIn:

op(

octets: OctetString;

destinations: List [Destination]; rank: Integer;



way: Way;

timeAdjust: Integer)

throws TransferInFailed;

単一のトランスファグループに組み合わされた1以上のトランスファユニットをエンジンに配送する。トランスファユニットはオクテット、デスティネーションおよびランクによって表される。ウェイバック（アーギュメント "way"）は、オペレーション "transferOut" を遂行したエンジンのテレネームを含む。ウェイバックは、戻りトリップを行なうために使用しうる既存のコネクションミーンズを提供しうる。

【2691】アーギュメント "timeAdjust" は、ソースプラットフォームのクロックがデスティネーションプラットフォームのクロックに遅れる秒の数である。レスポンドは、ソースエンジンのクロックに対して生成されたある時間を調整するためにこの情報を使用することができる。

【2692】エンジンがトランスファユニットを受けることができない場合には、例外 "TransferInFailed" が投出される。

【2693】transferFailed:

op(

フェイラ: Failure;

オクテット: OctetString;

デスティネーションズ: List [Destination];

ランク: Integer;

ウェイ: Way;

タイムアウト: Integer)

throws TransferFailedFailed;

単一のトランスファグループに組み合わされた1以上のフェイルズがトランスファユニットをエンジンに配送する。このオペレーションのためのパラメータは、オペレーション "transferIn" のものと同じであるが、フェイリヤの理由が付け加えられている（アーギュメント "failure"）。トランスファグループのためのソースエンジンにおいて "transferOutComplete" が受け入れられる前にオペレーション "transferFailed" が生じ得ることは可能である。

【2694】フェイリヤが "invalidMeans" であればレスポンドは可能ならばソースエンジンとなる。

【2695】フェイルしたトランスファユニットをレスポンドが受けることができない場合には、例外が投出される ("transferFailedFailed")。

【2696】transferOutFailed:

op(

理由: Failure;

グループ: Integer;

フェイラズ: List [Integer] [Nil]

unknownTransferGroup, UnknownTransferUnitを送出する;

トランスファグループ中の1以上のトランスファユニット（アーギュメント "groupID"）がフェイルしている。フェイリヤの原因はアーギュメント "リーズン" に与えられる。フェイルした特別のユニットは、オペレーショントランスファアウトの遂行において与えられた目的点のオリジナルリストへのインテジャインデックスのリスト（アーギュメント "failure"）によって特定される。このリストは1で始まり、この1は、トランスファユニットの最初のユニットを特定する。

【2697】リストが与えられない場合すなわちアーギュメント "failure" の値が0の場合、グループ中の各々のトランスファはフェイルしている。これはオリジナルグループのその他のユニットを列挙するリストに相当する。レスポンドがアーギュメント "groupID" ("UnknownTransferGroup") によって表されたオペレーション "transferOut" を以前に遂行しなかった場合、または、アーギュメント "failure" のアイテムの1つがこのトランスファグループのためのデスティネーションリストへの適切なインデックスでない場合 ("UnknownTransferUnit") 例外が投出される。

【2698】transferOutComplete:

op(groupID: Integer; serialNumber:

Integer) throws UnknownTransferGroup;

特別のトランスファグループ（アーギュメント "groupID"）に組み合わされたオペレーション "transferOut" が完成しなかったことを示す。特に、このグループのための別の "transferOutFailed" オペレーションはありえない。このグループ中のトランスファユニットに組み合わされたいかなるリソースもリリースすることができる。

【2699】アーギュメント "groupID" ("UnknownTransferGroup") によって表されるオペレーション "transferOut" をレスポンドが以前に遂行しなかった時は、例外が投出される。

【2700】4. 2 プラットフォーム (Platform)

オブジェクト

・Platform

クラス

Platform:

インスタンス = (...);

Construction

initialize

op(エンジン: Telename);

名前付けされたエンジン（アーギュメント "engine"）によって使用されるレスポンドをイニシャライズする。

【2701】Public Instance Operations

transferOut:

op(オクテット:

デスティネーション: List [Destination];

ランク: Integer;  
 way: Way|Nil;  
 desiredDuration: Integer;  
 maximumDuration: Integer;  
 グループ ID: Integer  
 InvalidDestination, InvalidWayを送出する;  
 アーギュメント "octets", "destinations" および "rank" によって定義されたトランスファグループを 1 以上のデスティネーションエンジンに配送することを要求する。グループはグループ id (アーギュメント "groupId") によって特定される。このオペレーションは、このグループを構成するトランスファユニットに "transferIn", "transferOutFailed" または "transferFailed" が後にリファレンスすることに結果する。

【2702】ランク (アーギュメント "rank") は、このトランスファが特権を持つか否かを特定する。

【2703】アーギュメント "way" が 0 でない場合にはトランスファをルーティングするために使用される。

【2704】トランスファの望ましい持続期間は、秒 (アーギュメント "desiredDuration") によって特定される。これはトランスファについて望ましい最大の持続時間である。望ましい持続時間の値が 0 であることは、このトランスファに望ましい持続時間が組み合わされていないことを意味する。

【2705】トランスファのための最大の許容持続時間は秒で特定される (アーギュメント "maximumDuration")。これはトランスファについて許可された最大時間である。オペレーション "transferOut" の後前記秒数の間オペレーション "transferOutComplete" が起きなかった場合の効果は、このグループについてオペレーション "failTransfer" が遂行されたかのような効果となる。最大持続時間の値 0 は、このトランスファに最大持続時間が組み合わされていないことを意味する。

【2706】デスティネーションリストが不適切であるか ("invalidDestination") アーギュメント "way" が供給されたら、0 でないアトリビュートを持たない場合 ("invalidway") 例外が投出される。

【2707】failTransfer:  
 op(groupId: Integer)  
 throws UnknownTransferGroup;  
 グループ id (アーギュメント "groupId") によって特定されたトランスファグループ中の全ての継続中のトランスファユニットがフェイルすることをリクエストする。

【2708】あるトランスファグループ中のフェイルする全部のトランスファユニットでないこともありうる。このオペレーションの結果としてフェイルしたトランスファユニットは、1 以上のオペレーション "transferOutFailed" とそれに続く "transferComplete" とを生ず

るであろう。

【2709】前回の "transferOut" においてグループ id が使用されなかった場合 (オペレーション "UnknownTransferGroup") 例外が投出される。

【2710】makeReservation  
 op(持続時間: Integer) Reservation  
 throws CannotReserve;

新しい持続時間 (アーギュメント "duration") 秒のリザーベーションをリターンする。リザーベーションは次のオペレーション "transferOut" のために留保可能な手段においてリザーベーションが使用される場合、活性となる。

【2711】リザーベーションが終了した時、リザーベーションがもはや有効でなくなり、そのリザーベーションを使用することを試みるトランスファはフェイルする。リザーベーションは、もはや必要とされない時に、オペレーション "resetReservation" によってキャンセルすべきである。プラットフォームは未使用のリザーベーションに対して最大のライフタイムを課することができる。

【2712】レスポンドがリザーベーションを供与できない時は例外 "CannotReserve" が投出される。

【2713】resetReservation:  
 op(リザーベーション: Reservation, duration: Integer) throws UnknownReservation, InvalidReservation;  
 現存のリザーベーション (アーギュメント "reservation") の持続時間は秒の持続時間に変更される。このオペレーションが呼び出された時にリザーベーションが活性である場合の効果は、新しい持続時間のリザーベーションとともにオペレーション "transferOut" がまさに行なわれたかのような効果となる。

【2714】持続時間を 0 にセットするとリザーベーションはキャンセルされる。キャンセルされたリザーベーションはもはや有効ではなくなる。リザーベーションが以前のオペレーション "makeReservation" ("unknownReservation") によって得られたものでないかまたはリザーベーションがもはや有効でない ("invalidReservation") 例外が投出される。

【2715】4. 3 デスティネーション (Destination)

オブジェクト  
 ・ Destination

クラス

Destination:

インスタンス = (...);

Construction

initialize

op(

ネーム: Telename|Nil;

アドレス: Teleaddress|Nil;

データ: OctetString|Nil;

リスポンダの固有の属性を同じ名前のアーギュメント  
(アーギュメント "name"、"address" および "data") とする。

【2716】あるデスティネーションの全ての属性が0  
であれば、デスティネーションは不適切である。

【2717】Private Instance Attributes

address:

アドレス|Nil;

デスティネーションのアドレス。

data:

テキストストリング |Nil;

トランスファユニットに組み合わされたエンジン特有のデータ。

name:

アドレス|Nil;

デスティネーションのネーム。

【2718】4. 4 フェイラー (Failure)  
オブジェクト

・Failure

クラス

Failure:

int failure = (···);

Private Instance Attributes

reason:

整数 (インデックス)

トランスファのフェイリヤの理由。この理由はテーブル  
C. 1 に示されたインデックスの列記として表される。

【2719】

【表75】

表 C. 1		
1	"invalidDestination"	invalidDestinationが用意される
2	"invalidWay"	invalidWayが用意される
3	"invalidMeans"	invalidMeansが用意される
4	"invalidReservation"	invalidReservationが用意される
5	"noDestinations"	destinationsが特定化されない
6	"maximumDurationExceeded"	最大期間が過ぎている
7	"transfersFailed"	failtransfersが呼ばれる
8	"noLocalResources"	プラットフォームがある供給源からの情報を得ることに失敗した
9	"destinationUnreachable"	transferユニットは配送しない
10	"transferInFailed"	transferInがリモートエンジンで動作不良
11	"securityViolation"	セキュリティ違反があった
12	"possibleDuplicate"	配送される可能性あり

プラットフォームは余分のフェイリヤタイプを供与し  
る。

【2720】5. ルーティング (ROUTING)  
あるトランスファユニットのオペレーション "transfer  
Out" は典型的に異なったエンジンにおいての同一のユ  
ニットのオペレーション "transferIn" に結果する。ト  
ランスファはソースプラットフォームにおいてフェイル  
した場合、トランスファユニットは、ソースエンジンに  
おいてのオペレーション "transferOutFailed" ととも  
にフェイルする。トランスファユニットはソースプラ  
ットフォームから成功の内にトランスファされたが、デ  
スティネーションエンジンにおいての成功したオペレ  
ーション "transferIn" open に加わらない場合、それ  
は、"transferFailed" open とともにエンジンに対  
してフェイルする。

【2721】トランスファグループの "transferOut"  
open は、コンポーネントトランスファユニットのN個  
の "transferOut" open と論理的に同等である。望ま  
しいランクの値、最大持続時間およびウェイは、全て  
のトランスファについて同一である。デスティネーション  
のリストは、open "transferIn" を含むことができ、

1以上のデスティネーションエンジンにおいて生ずる。

【2722】典型的には、あるプラットフォームは、あ  
るグループ中のデスティネーションのリストを検査し、  
デスティネーションエンジンにおいて "transferIn" o  
pn の最小の数を持ったユニットを供給しようと試  
みる。これは同一のエンジンに予定されたトランスファユ  
ニットが、1つのグループとして搬送され配送されるこ  
を意味する。

【2723】Routing Algorithm

"transferOut" open の遂行においてあるウェイが  
与えられた場合、プラットフォームはそのウェイを、ト  
ランスファのルーティングのために使用する。ウェイが  
与えられなかった場合プラットフォームはデスティ  
ネーションエンジンを定めなければならない。トランス  
ファユニットに組み合わされたネームおよびアドレスに  
従ってデスティネーションエンジンを定めるためのス  
テップは次の通りである。

【2724】<start>

名前が存在するがアドレスが存在しない場合

<use the name>

アドレスが存在するがネームが存在しない場合、プロバ

イダが現在の領域である場合

対応するネームを見いだすかまたは<フェイル>

デスティネーションのネームアトリビュートをネームにセットし

<start>

さもなければ

ネームとアドレスがともに存在していれば、

プロバイダが現在の領域である場合

<use the name>

さもなければ

<use the address>。

【2725】<use the name>:

オーソリティが現在の領域である場合、

アイデンティティからデスティネーションエンジンを定めるかまたは<フェイル>。

【2726】さもなければ

ロケーションからデスティネーションエンジンを定めまたは<フェイル>。

さもなければ

オーソリティのアウトポストを含むエンジンにルーティングするかまたは<フェイル>。

【2727】<use the address>

プロバイダが知られていれば

プロバイダのアウトポストを含むエンジンにルーティングする。

【2728】さもなければ

ルーティングアドバイス中の各々のプロバイダについてそれが現在の領域であれば

それをルーティングアドバイスから除去する

さもなければ

プロバイダが知られていれば

プロバイダのアウトポストを含むエンジンにルーティングする。

【2729】もうルーティングがなければ、<フェイル>。

【2730】<fail>:

プロバイダが現在の領域であれば

トランスファをフェイルする。

【2731】さもなければ

デフォルトエンジンがある場合

それにルーティングする。

【2732】さもなければ

トランスファをフェイルする。

【2733】6. C++におけるインターフェース (INTERFACE IN C++)

このセクションはエンジンプラットフォームインタフェースのC++インプリメンテーションを記述する。インタフェースはセクション4のテレ스크リプト仕様に由来する。

【2734】ここに提示されるインタフェースは、必然

的に現在のテレ스크リプトエンジンのインプリメンテーションのいろいろな様相を反映している。特にC++ APIは、族インタフェースに表れないエンジン上のオペレーションを含む。

【2735】Interface description

エンジンオブジェクトはC++クラス"CI"によって表される。エンジンオブジェクトに対するオペレーションは、CIのメンバ機能として供与される。

【2736】プラットフォームオブジェクトはCIのサブクラスによって表される。プラットフォームのプロバイダはCIのサブクラスであり、プラットフォームオペレーションを表すCIの仮想的な機能をインプリメントする。

【2737】エンジンとプラットフォームとの両方に対するオペレーションにおいてパラメータとして表れるテレ스크リプトオブジェクトは、テレ스크リプトオブジェクトの限定された表現をインプリメントするある数のC++クラスによって表される。例えば、"TsInteger"がAPIに表れる時にはいつでもテレ스크リプト"Integer"を表すクラス"TsInteger"がある。これらのテレ스크リプトインタフェースオブジェクトの完全な説明はこのアペンディックスの次のセクションにおいてなされる。

【2738】インタフェースのテレ스크リプト表現において特定される例外は、列記された形式としてAPIにおいてインプリメントされ、この列記された形式は、エンジンおよびプラットフォームの両方の全てのオペレーションからリターンされる。

【2739】エンジンとプラットフォームとの間に交わされたデータエレメントのストレージは、オペレーションの呼び出し者によって所有される。ストレージはオペレーションが完全になされた時に再請求しうる。これに対する唯一の例外はオペレーション"transferOut"である。エンジンは対応するオペレーション"transferOutComplete"が遂行され終わるまではトランスファアウトのエレメントを再請求しない。

【2740】Tasks

エンジンは、非先取(non-preemptive)スケジューラとともに別のスレッドとして走る多重の共働タスクとしてインプリメントされる。CIクラスの1つの例は、エンジン中の1つのスレッドとして走らせる。クラスCIのサブクラスは、ファンクション"main0"を供与し、このファンクションは、1度呼び出され、プラットフォーム上に余分なオペレーションがまだ指示されていない場合にはリターンされねばならない。

【2741】"main0"は、継続中のオペレーション"transferIn"がない場合にエンジンスケジューラにイールド(yield)することが期待される。"main0"は"wait0"を呼び出すことによってこれを行なう。"wait0"はCIがそのためにイールドすることを意図し

ているミリ秒の数であるアーギュメントを取る。典型的にはエンジンは少なくともその時間量が経過するまではC Iのタスクを再スケジュールしない。"wait 0"に対する0のアーギュメントは、C Iがイールドしているがエンジンスケジューラによって可及的早く再スケジュールされることを望むことを意味している。

【2742】"wait 0"へのアーギュメント-1は、C Iが不特定にブロックすることを望んでいるという意味を持っている。この場合C Iは、C Iが再スケジュールされることを望むならば、エンジン上に"satisfy 0"を呼び出すことが期待されている。"satisfy 0"は、エンジンがスレッドをスケジュールすると直ちに、ウェイトにブロックされたスレッドの実行を再開させる。この後者のスキームは、新しいコミュニケーションのリクエストがなかった場合に非同期的なイベントとして供与されたC Iインプリメンテーションにおいて適切であろう。

【2743】ある動作中のシステムにおいては、C Iインプリメンテーションは、エンジンスケジューラにイールドすることを望み、ファイルデスクリプタが準備状態になった時に再スケジュールされることを望むかもしれない。オペレーション"waitfd"は、待機されるファイルデスクリプタを特定し、タイムアウトとともに多重ファイルデスクリプタについてセレクトするために、"wait 0"とともに使用されうる。

【2744】open "transferOut"の遂行は、呼び出したテレスク립トプロセスのスレッドにおいて実行されるのであり、C Iタスクのスレッドとして行なわれるのではない。

【2745】エンジンおよびプラットフォームによるプラットフォームは、必要ならばその後にスケジュールされるワークの待ち行列を作る(queueing)ことによって比較的速やかに実行されねばならない。エンジンとプラットフォームの両方は長時間にわたってブロックすべきではない。

【2746】極端な場合には、オペレーション"transferOut"の量がC Iによって供与されたバッファリングを上回ることがある。

【2747】CI クラス

Construction

CI:

CI(TsTelename \*name);

新しいC Iオブジェクトを構成する。"name"はこのC Iを使用するエンジンのテレネームである。

【2748】-CI:

仮想的 -CI 0

このC Iをシャットダウンし破壊する。

【2749】main:

仮想的なlong main 0=0;

C Iのメインループ。これは構成後ある時間がしてから

呼び出される。C Iは、このファンクションの実行の間、例えば"wait 0"とともにエンジンに対してイールドすべきである。

【2750】Engine Operations

transferIn:

TsStatus transferIn (TsOctets \*bagOfBits, TsDestinationList \*destList, unsigned int rank, TsWay \*wayBack, int

timeAdjust);

1以上のトランスファユニットをエンジンに移送する。

【2751】transferFailed:

TsStatus transferFailed (TsFailure failure, TsOctets \*bagOfBits,

TsDestinationList\* destList,

unsigned int rank,

TsWay \*wayBack, int timeAdjust);

1以上のフェイルしたトランスファユニットをエンジンにトランスファする。

【2752】transferOutFailed:

TsStatus transferOutFailed (TsFailure failure, unsigned int groupId, TsIntegerList \*list);

このエンジンからのopen "transferOut"の以前の実行においてフェイルした1以上のトランスファユニットを特定する。

【2753】transferOutComplete:

TsStatus transferOutComplete (unsigned int groupId, unsigned int transferSN);

トランスファグループ(アーギュメント"groupId")のトランスファが完成したことを表す。このグループ中の全てのトランスファユニットに組み合わせられたストレージは再請求しうる。

【2754】Platform Operations

transferOut:

仮想的なTsStatus transferOut (TsOctets \*bagOfBits, TsDestinationList \*destList,

Unsigned int rank, TsWay \*wayOut,

unsigned int desiredDuration,

unsigned int maximumDuration,

unsigned int groupId)=0;

このトランスファグループ中のトランスファユニットをそれぞれの目的点に転送する(アーギュメント"destList")。

【2755】failTransfer:

virtual TsStatus failTransfer (unsigned int groupId)=0;

トランスファグループ(アーギュメント"groupId")に組み合わせられた全てのトランスファユニットをフェイルさせる。

【2756】makeReservation:

virtual TsStatus makeResevation (unsigned int dura

tion, TsReservation\*reservation)=0;  
指定された持続時間（アーギュメント”duration”）とともに新しいリザベーション（アーギュメント”reservation”）をリクエストする。

【2757】resetReservation:  
virtual TsStatus resetReservation (TsReservation reservation, unsignedint duration)=0;  
リザベーションの持続時間を新しい持続時間（アーギュメント”duration”）にリセットする。持続時間0は、リザベーションをキャンセルする効果を持つ。

【2758】Task Operations  
wait:  
void wait (long msec);  
”msec”ミリ秒の間現在のエンジンスロットをイールドする。この機能は、このスレッドが再スケジュールされた時にリターンされる。アーギュメント”msec”が0であれば、エンジンスケジューリングポリシーが許容する限り早くスレッドを再スケジュールする。アーギュメント”msec”が-1であれば、”satisfy0”（下記参照）が呼び出されるかまたは”waitfd0”と指定されたファイルディスクリプタが準備状態となるまで、”waitfd”へのこの呼び出しはリターンされない。

【2759】satisfy:  
void satisfy0;  
CIスレッドが呼び出し”wait0”においてブロックされた場合、CIスレッドを再スケジュールする。

【2760】waitfd:  
void waitfd (int fd, short mask);  
”mask”によって指定された読み込みまたは書き込みのためにファイルディスクリプタ”fd”が準備状態になった場合、後のコール（呼び出し）”wait0”がリターンされることを指定する。”mask”は、ファイルディスクリプタがそれぞれ読み込み可能および書き込み可能であることを選択する値”POLLIN”および”POLLOUT”のORを取ることによって構成されたビットマスクである。”mask”が0であることは、このファイルディスクリプタがオペレーション”wait0”にもはや影響すべきでないことを意味する。

【2761】waitresult:  
short waitresult (int fd);  
前回の呼び出し”wait0”の間にファイルディスクリプタ”fd”が準備段階になっているか否かをテストするために使用しうるビットマスクをリターンする。その結果は、ビット”POLLIN”および”POLLOUT”が”waitfd0”において指定されていればこれらのビットを有するビットマスクがセットされ、ファイルディスクリプタがそれぞれリードおよびライトについて準備状態となることである。

【2762】stop:  
void stop0;

このCIについて全ての”transferOut”openをエンジンによってスロットルさせる。open”go”または”send”を実行しているテレスクリプトエージェントはブロックさせうるようなそれ以上のトランスファはこのCIにはリクエストされない。CIは別のトランスファを待ち行列とすることによってCIによるデータを失った場合にこのオペレーションを引き起こすべきである。

【2763】start:  
void start0;  
もはやopenをスロットルすべきでないことをエンジンに通報する。

【2764】7. テレスクリプトのインターフェースオブジェクト（TELESCRIPTINTERFACE OBJECTS）  
外部システムすなわち他のテレスクリプトエンジンまたはテレスクリプトに基づかない通信システムとの通信を可能にするため、テレスクリプトエンジンは、プラットフォームまたはテレスクリプトエンジン自身に内在するオペレーティングシステムによって供給される通信サービスに依存しなければならない。テレスクリプトエンジンは、これらのサービスにアクセスするために、情報がテレスクリプトアブストラクションに流入したりそれから流出したりすることを可能とするためのメカニズムを指示していなければならない。詳細には、このメカニズムは、内在するコミュニケーションサービスにとって理解可能なこれらのオブジェクトの対応する表現とキーテレスクリプトオブジェクトとの間の変換を可能とするものでなくてはならない。

【2765】テレスクリプトエンジンインタフェースクラスは、このニードを充たすC++クラスのセットのためのアブストラクトデフィニション（抽象的な定義）を供与する。この供与される定義は、C++クラスが十分に定義されていなく、テレスクリプトエンジンの機能性についての最初の要求の言葉で記述されているため抽象的である。

【2766】これらのクラスの内部的な機構および内在する通信サービスとの関係について完全な自由がこれらのクラスのインプリメンターに任せられている。

【2767】7. 1 インターフェースコンベンション（Interface Conventions）  
テレスクリプトエンジンインタフェースクラスのインプリメンテーションは、クラスインタフェースを通過するデータアイテムの使用についての1組みのコンベンションに従わねばならない。これらのコンベンションは、オーナーシップの規則の言葉で表現され、ここにオーナーシップは、ある与えられたデータピースを変更する権利およびそれを破壊する責任とともに定義される。

【2768】他に指定がなければ、テレスクリプトエンジンインタフェースクラスのインプリメンテーションは、次のインタフェースコンベンションに従わねばなら

ない。

【2769】リファレンスまたはポインタアーギュメントを受け入れる全ての機能に対して、参照されまたは指示されたデータのオーナーシップはコーラー（呼び出した者）から呼び出されたオブジェクトに移行し、そのためオブジェクトは、適切な時にデータを変更する権利およびそのデータを破壊する責任を持つ。

【2770】リファレンスまたはポインタの値をリターンする全ての機能に対して、参照または指示されたデータのオーナーシップは、呼び出されたオブジェクトによって保持される。しかしオブジェクトは、オブジェクトを変更しうるファンクションへの次のコールがなされるまでは参照され/指示されたデータが勝手に変えられないことを保証する。

【2771】オブジェクトは破壊された時それがオーナーシップを有する全てのデータを破壊する。

【2772】7. 2 Tsキャラクタ (TsCharacter)

"TsCharacter" タイプのデフィニションは、この開示のアペンディックスAに定められたように、テレクリプトキャラクタクラスの限定されたC++表現を供与する。

【2773】"TsCharacter" タイプのインプリメンテーションは、全てのASCIIキャラクタを収容することができることが望まれている。これは最初の要求である。好ましくは、"TsCharacter" タイプのインプリメンテーションは全てのユニコードキャラクタを含むことができる。

【2774】7. 3 Tsデスティネーション (TsDestination)

"TsDestination" クラスはこのアペンディックスのセクション4に規定されたように、テレクリプトデスティネーションクラスのC++表現を供与する。"TsDestination" クラスのインプリメンテーションは、最小で、次の機能性を供与するものとする。

【2775】クラス TsDestination

```
(
public:
TsDestination(TsTeleaddress*teleaddress, TsTelenam
e * telename, TsOctets * data);
-TsTeleaddress();
const TsTeleaddress * getTeleaddress()const; const
TsTelename * getTelename() const; const TsOctets
* getData() const;);
Construction
TsDestination:
TsDestination (TsTeleaddress * teleaddress, TsTele
name * telename, TsOctets * data)
は、新しい"TsDestination" オブジェクトを構成す
る。
```

【2776】teleaddress

新しい"TsDestination" オブジェクトのテレアドレスの値を形成するべき"TsTeleaddress" オブジェクトへのポインタまたは、デスティネーションオブジェクトがテレアドレスの値を持たない場合、0ポインタ。

【2777】telename

新しい"TsTelename" オブジェクトのテレネーム値を形成するべき"TsDestination" オブジェクトへのポインタまたは、デスティネーションオブジェクトがテレネームの値を持たない場合、0ポインタ。

【2778】data

新しい"TsDestination" オブジェクトのデスティネーションオブジェクトの値を形成するために"TsOctets" オブジェクトへのポインタか、または、デスティネーションオブジェクトがデスティネーションデータの値を持たない場合、0ポインタ。

【2779】Destruction

-TsDestination:

-TsDestination()

"TsDestination" オブジェクトを破壊する。"TsDestination" オブジェクトと、そのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。

【2780】Data Access

getTeleaddress:

```
const TsTeleaddress*getTeleaddress()const
```

"TsDestination" オブジェクトのテレアドレスの値へのポインタをリターンするかまたは、デスティネーションオブジェクトがテレアドレスの値を持たない場合、0ポインタをリターンする。

【2781】getTelename:

```
const TsTelename*getTelename()const
```

"TsDestination" オブジェクトのテレネームの値へのポインタをリターンするか、または、デスティネーションオブジェクトがテレネームの値を持たない場合、0ポインタをリターンする。

【2782】getData:

```
const TsOctets * getData()const
```

"TsDestination" オブジェクトのデスティネーションデータの値へのポインタをリターンするか、または、デスティネーションオブジェクトがデスティネーションデータの値を持たない場合、0ポインタをリターンする。

【2783】7. 4 Tsデスティネーションリスト

(TsDestinationList)

"TsDestinationList" クラスは、クラス"destination" のオブジェクトを収容するように拘束された、この開示のアペンディックスAに規定されたテレクリプトクラス"List" のC++表現を供与する。"TsDestinationList" クラスのインプリメンテーションは最小でも次の機能性を供与しなければならない。

【2784】{

```
public:
TsDestinationList 0;
-TsDestinationList 0;
const TsDestination*nextDestination 0;void reset 0; size_t destinationCount 0 const;
void appendDestination(TsDestination*destination);
};
```

Construction

TsDestinationList:

TsDestinationList 0

エンプティの”TsDestinationList”オブジェクトを構成する。

【2785】Destruction

-TsDestination-List:

-TsDestinationList 0

”TsDestinationList”オブジェクトを破壊する。”TsDestinationList”オブジェクトおよびそのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。

【2786】Data Access

appendDestination:

void appendDestination(TsDestination\*destination)

”TsDestination”オブジェクトの形の単一のデスティネーションを、”TsDestinationList”オブジェクト内に含まれるデスティネーションのリストに付加する。

【2787】destination

”TsDestination”オブジェクト中のデスティネーションのリストに付加されるべきデスティネーションを伝える”TsDestination”オブジェクトへのポインタ。

【2788】nextDestination:

const TsDestination\*nextDestination 0

”TsDestination”オブジェクトの形のデスティネーションへのポインタをリターンするか、または0ポインタをリターンする。”nextestination 0”機能は多重の呼び出しを通じて、”TsDestinationList”オブジェクトに含まれる1組のデスティネーションを、それらが収容されている順序でリターンする。最後のデスティネーションがリターンされた後、リセットファンクションの次の呼び出しまで、”nextestination 0”機能は0ポインタをリターンする。

【2789】reset:

void reset 0

”nextestination 0”機能の次の呼び出しによって、”TsDestinationList”オブジェクト中に含まれる最初のデスティネーションをリターンさせる。

【2790】destinationCount:

size\_t destinationCount 0 const

”TsDestinationList”オブジェクト中に現在含まれるデスティネーションの数をリターンする。

【2791】7. 5 Tsエグジスティングコネクショ

ンミーンズ (TsExistingConnectionMeans)

”TsExistingConnectionMeans”クラスは、このアペンディックスのセクション3に規定されたように、テレスクリプト”TsExistingConnectionMeans”クラスのC++表現を供与する。”TsExistingConnectionMeans”クラスのインプリメンテーションは最小でも次の機能を提供しなければならない。

【2792】クラス TsExistingConnectionMeans:public TsReservableMeans

{

public: TsExistingConnectionMeans

(const TsReservation&reservation, TsOctets\*connectionId);

-TsExistingConnectionMeans 0;

const TsOctets \* getConnectionId

0 const; };

Construction

TsExisting-ConnectionMeans:

TsExistingConnectionMeans(const TsReservation& reservation, TsOctets \*connectionId)

新しい”TsExistingConnectionMeans”オブジェクトを構成する。

【2793】reservation

新しい”TsExistingConnectionMeans”オブジェクトの”TsReservableMeans”スーパークラスコンポーネントのリザーベーションの値を形成するべき”TsReservation”オブジェクトへのリファレンス。

【2794】connectionId

新しい”TsExistingConnectionMeans”オブジェクトのコネクションの識別子の値を形成するべき”TsOctets”オブジェクトへのポインタ。

【2795】Destruction

-TsExisting-ConnectionMeans:

-TsExistingConnectionMeans 0

”TsExistingConnectionMeans”オブジェクトを破壊する。”TsExistingConnectionMeans”オブジェクトおよびこのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。

【2796】Data Access

getConnectionId:

const TsOctets \* getConnectionId 0

const

”TsExistingConnectionMeans”オブジェクトに含まれるコネクション識別子の値へのポインタがリターンされる。

【2797】7. 6 Tsインテジャ (TsInteger)

”TsInteger”タイプのデフィニションは、テレスクリプトクラス”Integer”の限定されたC++表現を与え



る。

【2798】"TsInteger"タイプのインプリメンテーションは、範囲-2147483648から+2147483647の範囲にある値を保持することができなければならない。

【2799】7. 7 インテジャリスト (TsIntegerList)

"TsIntegerList" クラスは、クラス"integer"のオブジェクトを収容するように拘束された、この開示のアペンディックスAに定義されたテレascriptクラス"List"のC++表現を供与する。"TsInteger"クラスのインプリメンテーションは最小でも次の機能を供与しなければならない。

【2800】この場合には、CIは、"stop()"を呼び出し、これは、open "transferOut"へのその後の全ての呼び出しを阻止する効果を持つであろう。"start()"への後の呼び出しは、出力を取り、ブロックの結果として停止されねばならなかったエンジンタスクをおそらくは再スケジュールする。

【2801】クラス TsIntegerList

```
|
public: TsIntegerList();
-TsIntegerList();
const TsInteger*nextInteger();
void reset();
size_t integerCount() const;
void appendInteger(TsInteger Integer);};
Construction
```

TsIntegerList:

TsIntegerList()

新しい"TsIntegerList"オブジェクトを構成する。

【2802】この構成の後"appendInteger()"機能が呼び出されるまでに、新しく作り出されたインテジャリストは空となる。

【2803】Destruction

-TsIntegerList:

-TsIntegerList()

"TsIntegerList"オブジェクトを破壊する。

【2804】"TsIntegerList"オブジェクトおよびそのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。

【2805】appendInteger;

void appendInteger(TsInteger integer)

"TsIntegerList"オブジェクトの形の単一のインテジャを、Tsインテジャリストオブジェクトに含まれたインテジャのリストに付加する。

【2806】integer

"TsIntegerList"オブジェクトに含まれるインテジャのリストに付加されるべきインテジャの値。

【2807】nextInteger:

const TsInteger \* nextInteger()

Tsインテジャリスト/オブジェクトの形のインテジャ(整数)へのポインタをリターンするかまたは0ポインタをリターンする。"nextInteger()"機能は多数回の呼び出しを通じて、"TsIntegerList"オブジェクトに含まれるインテジャのセットを、それらが収容されている順序でリターンする。最後の整数がリターンされた後、リセットファンクションが次に呼び出されるまでに"nextInteger()"機能によって、0ポインタをリターンする。

【2808】reset:

void reset()

"nextInteger()"の次の呼び出しによって、"TsIntegerList"オブジェクトに含まれる最初の整数をリターンさせる。

【2809】integerCount:

size\_t integerCount() const

"TsIntegerList"オブジェクトに現在含まれる整数の数をリターンする。

【2810】7. 8 Tsミーンズ (TsMeans)

"TsMeans"クラスは、この開示のアペンディックスAに規定されたテレascriptクラス"Means"のC++の表現を供与する。"TsMeans"のインプリメンテーションは、最小でも次の機能を供与しなければならない。

【2811】クラス TsMeans

```
|
public:
typedef enum
|
TsMeansType_PSTN=0,
TsMeansType_ExistingConnection=1, }Type;
virtual -TsMeans()=0;
Type type() const;
protected:
TsMeans (Type type);
|;
Construction
TsMeans;
TsMeans (Type meansType)
```

新しい"TsMeans"オブジェクトを構成する。

【2812】meanType

新しい"TsMeans"オブジェクトのミーンズ形の値を指定する列記。

【2813】Destruction

-TsMeans:

仮想的な -TsMeans()

"TsDestination"オブジェクトを破壊する。"TsDestination"オブジェクトとそのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。

【2814】Data Access

type:

Type type 0 const

"TsMeans" オブジェクトのミーンズ形の値をリターンする。

【2815】7. 9 Tsオブジェクトスペシファイヤ (TsObjSpecifier)

"TsObjSpecifier" は、各々が1つのプロトコルを表すオブジェクトのクラスのC++表現を与え、このプロトコルを他のプロトコルから識別する。一例として、"TsObjSpecifier" によって表されるクラスの1つのメンバは、セキュリティを与える特定の1つの方法を表すように使用することができる。"TsObjSpecifier" クラスのインプリメンテーションは、最小でも、次の機能を供与しなければならない。

【2816】クラス TsObjSpecifier

```
|
public:
TsObjSpecifier(TsInteger objId,
TsOctets * namingAuthority);
-TsObjSpecifier();
TsInteger getObjId() const;
const TsOctets * getNamingAuthority() const; };
Construction
TsObjSpecifier:
TsObjSpecifier(TsInteger objId, TsOctets * namingAuthority)
新しいTsオブジェクトスペシファイヤオブジェクトを構成する。
```

【2817】objId

新しい"TsObjSpecifier" オブジェクトのオブジェクト識別子の値を形成するべき整数の値。

【2818】naming Authority

新しい"TsObjSpecifier" オブジェクトのネーミングオーソリティを形成するべき"TsOctets" オブジェクトへのポインタか、または、新しい"TsObjSpecifier" オブジェクトがネーミングオーソリティの値を含まない場合、0ポインタ。

【2819】Destruction

```
-TsObjSpecifier:
-TsObjSpecifier()
"TsObjSpecifier" オブジェクトを破壊する。"TsObjSpecifier" オブジェクトとそのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。
```

【2820】Data Access

```
getObjId:
TsInteger getObjId() const
"TsObjSpecifier" オブジェクトに含まれるオブジェクト識別子の値がリターンされる。
```

【2821】getNaming-Authority:

```
const TsOctets * getNamingAuthority() const
```

"TsObjSpecifier" オブジェクトに含まれるネーミングオーソリティの値を伝える"TsOctets" オブジェクトへのポインタをリアするか、または、"TsObjSpecifier" オブジェクトがネーミングオーソリティの値を含まない場合、0ポインタをリターンする。

【2822】7. 10 Tsオクテット (TsOctet)

Tsオクテットタイプがこの開示のアペンディックスAに規定されたテレスクリプトクラス"octet"のC++表現を供与する。

【2823】"TsOctets" タイプはC++符号なしcharとして定義される。

【2824】7. 11 Tsオクテツ (TsOctets)

"Octets" クラスは、この開示のアペンディックスAに規定されたテレスクリプトクラス"octets"のC++表現を与える。"TsOctets"のインプリメンテーションは、最小でも、次の機能を与えなければならない。

クラス TsOctets

```
|
public:
enum Copy Ownership {makeCopy, ownCopy, borrowCopy};
TsOctets(size_t size);
TsOctets(size_t size, const TsOctet * data, CopyOwnership ownership=makeCopy);
-TsOctets();const TsOctet*nextSeg(int&segSize);
void reset();
size_t size() const;
void appendSeg(size_t size, const TsOctet* data);
};
Construction
TsOctets:
TsOctets("size_t size, TsOctet*data, CopyOwnership ownership)
"TsOctet" オブジェクトの単一の列から"TsOctets" オブジェクトを構成する。
```

【2825】size

新しい"TsOctets" オブジェクトに含まれるオクテットの数。

【2826】data

新しい"TsOctets" オブジェクトに含まれるべきオクテットの列すなわち"TsOctet" オブジェクトへのポインタ。この列中のエレメントの数は、"サイズ"の値に等しいかまたはこれよりも大きくななければならない。

【2827】注:"サイズ"が0である場合、呼び差し者(コーラー)によって0の長さの列が供与されねばならない。

【2828】ownership

オクテットの列を新しい"TsOctets" オブジェクトに組み込む方法を特定した列記。可能な値は次の通りであ

る。

【2829】"makeCopy" : オクテットの列中のデータは、別の物理的なコピーの作成によって新しい"TsOctets" オブジェクトに組み込まねばならない。この構成に続いてコーラーは、元のオクテット列の完全なオーナーシップを保持している。

【2830】特定されたオーナーシップアーギュメントが"makeCopy"である場合、コーラーによって供与されたオクテット列は、静的に、動的にまたは自動的に割り当てられることができる。

【2831】"ownCopy" : オクテット列中のデータは、インプリメンテーションの裁量によって、供与されるオクテット列自身の組み込みを介して直接に新しい"TsOctets" オブジェクトに組み込むことができる。コーラーはこの構成に続いて、供与されたオクテット列のオーナーシップを新しい"TsOctets" オブジェクトに引き渡す。

【2832】特定されたオーナーシップのアーギュメントが"ownCopy"である場合、コーラーによって供与されたオクテット列は動的に割り当てられなければならない。

【2833】"borrowCopy" : オクテット列中のデータは、供与されたオクテット列自身の組み込みを介して、インプリメンテーションの裁量で、新しい"TsOctets" オブジェクトに直接に組み込むことができる。"TsOctets" オブジェクトが構成された後、その破壊まで、コーラーは、供与されたオクテット列自身の組み込みを介して、インプリメンテーションの裁量で、新しい"TsOctets" オブジェクトに直接に組み込むことができる。"TsOctets" オブジェクトが構成された後その破壊までコーラーは、供与されたオクテット列の列に対するコーラのオーナーシップを新しい"TsOctets" オブジェクトにローンする。この期間の間コーラも、"TsOctets" オブジェクトも、オクテット列を変更したり破壊したりしない。"TsOctets" オブジェクトが開始されたら、オクテット列の完全なオーナーシップは再びコーラに戻される。

【2834】特定されたオーナーシップのアーギュメントが"borrowCopy"である場合、コーラによって供与されたオクテット列は、静的に、動的にまたは自動的に割り当てることができる。

【2835】TsOctets:

TsOctets (size\_t eventualSize)

"appendSeg 0" 機能を用いて、データを付加する準備として、空の"TsOctets" オブジェクトを構成する。これについては以下に説明される。 eventualSize

新しい"TsOctets" オブジェクトに時に含められるべき最大数のオクテット。

【2836】Destruction

-TsOctets:

-TsOctets 0

"TsOctets" オブジェクトを破壊する。"TsOctets" オブジェクトおよびそのオブジェクトがオーナーシップが責任を持つ全てのデータは破壊される。

【2837】Data Access

appendSeg:

void appendSeg (size\_t size,

TsOctet \* data,

CopyOwnership ownership)

オクテット列すなわち"TsOctets" オブジェクトの列を、"TsOctets" オブジェクト中にすでに存在しているオクテットのセットに付加する。

【2838】注:"TsOctets" 以外のコンストラクタによって、すなわち、前記コンストラクタ機能のアーギュメント"イベンチュアルサイズ"によって、サイズ"イベンチュアルサイズ" コンストラクタによって、作り出された"TsOctets" オブジェクトに対してこの機能呼び出す効果は、インプリメンテーションによって規定される。さらにコンストラクタの"イベンチュアルサイズ" パラメータに宣言された以上のデータを付加する効果もインプリメンテーションによって規定される。

【2839】size

"TsOctets" オブジェクトに付加されるべきオクテットの数。

【2840】data

"TsOctets" オブジェクトに付加されるべきオクテットすなわち"TsOctet" オブジェクトの列へのポインタ。列中のエレメントの数は、"size" の値に等しいかこれより大きくなければならない。

【2841】注:"size" が0である場合、0の長さの列がコーラによって供与されねばならない。

【2842】ownership:

オクテット列を"TsOctets" オブジェクトに組み込む方法を特定する列記。可能な値およびその対応する意味は、"TsOctets (size\_t size, TsOctet \* data, CopyOwnership ownership)" 機能について説明したものと同一である。

【2843】nextSeg:

const TsOctet \* nextSet (size\_t & segSize)

オクテット列およびサイズの値または0ポインタをリターンする。"nextSeg" ファンクションは、多数回の呼び出しを通じて、"TsOctets" オブジェクトに含まれる1つの列をその出現の順序で一緒に伝えるオクテット列セグメントのコレクションをリターンする。その最後のセグメントのリターンの後、"reset" ファンクションが次に呼び出されるまでに、"nextSeg" ファンクションは、0ポインタをリターンし、"segSize" パラメータの値は変更しないままにしておく。

【2844】注：リターンされるセグメントの全数および各々の個別のセグメントのサイズはインプリメンテーションに依存する。

【2845】segSize

リターンされるセグメントの大きさとともに更新されるべき符号なしの整数の値へのリファレンス。

【2846】reset:

void reset()

”TsOctets”オブジェクト中に含まれる最初のオクテットのセグメントを”nextSeg”ファンクションの次の呼び出しによってリターンさせる。

【2847】size:

size\_t size() const

”TsOctets”オブジェクトに現在含まれるオクテットの全数をリターンする。

【2848】7. 12 TsPSTNミーンズ (TsPSTNMeans)

”TsPSTNMeans”クラスは、このアペンディックスのセクション3に規定されたテレクリプトクラス”PSTNMeans”のC++表現を供与する。”TsPSTNMeans”クラスのインプリメンテーションは最小でも次の機能を有しなければならない。

【2849】クラス TsPSTNMeans: public TsReservableMeans { public:

TsPSTNMeans(const TsReservation & reservation, TsTelenumber \* telenumber); ~TsPSTNMeans();

TsTelenumber \* getTelenumber() const; };

Construction

TsPSTNMeans:

TsPSTNMeans(const TsReservation & reservation, TsTelenumber \* telenumber)

新しい”TsPSTNMeans”オブジェクトを構成する。

【2850】reservation

新しい”TsPSTNMeans”オブジェクトの”TsReservableMeans”スーパークラスコンポーネントのリザーベーションの値を形成するべき”TsReservation”オブジェクトへのリファレンス。

【2851】telenumber

新しい”TsPSTNMeans”オブジェクトのテレナンバの値を形成するべき”TsTelenumber”オブジェクトへのポインタ。

【2852】Destruction

~TsPSTNMeans:

~TsPSTNMeans()

”TsPSTNMeans”オブジェクトを破壊する。”TsPSTNMeans”オブジェクトおよびそのオブジェクトがオーナーシップ責任を持つ全てのデータが破壊される。

【2853】Data Access

getTelenumber:

const TsTelenumber \* getTelenumber

() const

”TsPSTNMeans”オブジェクトに含まれるテレナンバの値をポインタにリターンする。

【2854】7. 13 Tsリザーバブルミーンズ (TsReservableMeans)

”TsReservableMeans”クラスは、このアペンディックスのセクション3に定義されたテレクリプトクラス”ReservableMeans”のC++表現を与える。”TsReservableMeans”クラスのインプリメンテーションは最小でも次の機能を与えなければならない。

【2855】クラス

TsReservableMeans: public TsMeans {

protected:

TsReservableMeans(TsMeans::Type type, const TsReservation & reservation); virtual ~TsReservableMeans();

public:

TsReservation & getReservation() const; };

Construction

TsReservableMeans:

TsReservableMeans(TsMeans::Type type, const TsReservation & reservation)

新しい”TsReservableMeans”オブジェクトを構成する。

【2856】type

新しい”TsReservableMeans”オブジェクトの”TsMeans”スーパークラスコンポーネントのミーンズ形の値を特定する列記。

【2857】reservation

新しい”TsReservableMeans”オブジェクトに含まれるべきリザーベーションの値を伝える”TsReservation”オブジェクトへのリファレンス。Destruction-TsReservableMeans:

仮想的な~TsReservableMeans()

”TsReservableMeans”オブジェクトを破壊する。”TsReservableMeans”オブジェクトおよびこのオブジェクトがオーナーシップの責任を持つ全てのデータを破壊する。

【2858】Data Access

getReservation:

const TsReservation & getReservation() const

”TsReservableMeans”オブジェクト内に含まれるリザーベーションの値へのリファレンスをリターンする。

【2859】7. 14 Tsリザーベーション (TsReservation)

”TsReservation”クラスはこのアペンディックスのセクション3に定義されたテレクリプトクラス”Reservation”のC++表現を与える。”TsReservation”クラスのインプリメンテーションは最小でも次の機能を与え

なければならない。

【2860】クラス TsReservation

```
{
public:
TsReservation(TsInteger id = 0);
-TsReservation 0;
TsInteger id 0 const;
};
Construction
TsReservation:
TsReservation (TsInteger id = 0)
新しい"TsReservation" オブジェクトを構成する。
```

【2861】id  
新しい"TsReservation" オブジェクトのリザーベーション識別子の値を形成するべき"TsInteger"。

【2862】Destruction

```
-TsReservation:
-TsReservation 0
"TsReservation" オブジェクトを破壊する。"TsReservation" オブジェクトおよびこのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。
```

【2863】Data Access

```
id:
TsInteger id 0 const
"TsReservation" オブジェクトのリザーベーション識別子の値をリターンする。
```

【2864】7. 15 Ts スtring (TsString)  
Ts スtring クラスは、この開示のアペンディックス A に定義されたテレスク립トクラス"String" の C++ 表現を与える。"TsString" クラスのインプリメンテーションは最小でも次の機能を与えなければならない。

Issue: This needs to be extended to support non-ASCII character sets.

クラス TsString

```
{
public:
TsString (const TsCharacter * string); -TsString 0;
const TsCharacter * getString 0 const; size_t size 0 const;
};
Construction
TsString:
TsString(const TsCharacter*string)
新しい"TsString" オブジェクトを構成する。
```

【2865】string  
新しい"TsString" オブジェクトに含まれるべきキャラクタの String を伝える、0 キャラクタによって終了される"TsCharacter" 列へのポインタ。

【2866】Destruction

```
-TsString:
-TsString 0
"TsString" オブジェクトを破壊する。"TsString" オブジェクトおよびそのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。
```

【2867】Data Access

```
getString:
const TsCharacter*getString 0 const
"TsString" オブジェクトに含まれるキャラクタ String へのポインタをリターンする。
```

【2868】size:

```
size_t size 0 const
"TsString" オブジェクトに現在含まれるキャラクタの数をリターンする。
```

【2869】7. 16 Ts テレアドレス (TsTeleaddress)

"TsTeleaddress" クラスは、この開示のアペンディックス A に定義されるテレスク립トクラス"TsTeleaddress" の C++ 表現を供与する。"TsTeleaddress" クラスのインプリメンテーションは最小でも次の機能を供与しなければならない。

【2870】クラス TsTeleaddress

```
{
public:
TsTeleaddress (TsOctets * provider, TsString * location);
-TsTeleaddress 0;
const TsOctets*getProvider 0 const; const TsString*getLocation 0 const;
const TsOctets*getNextRoutingAdvice 0; void reset 0;
int routingAdviceCount 0 const;
void appendRoutingAdvice (TsOctets*advice);};
Construction
TsTeleaddress:
TsTeleaddress (TsOctets*provider, TsString*location)
新しい"TsTeleaddress" オブジェクトを構成する。
```

【2871】この構成の後に、新しく作り出されたテレアドレスオブジェクト中に含まれるルーティングアドバイスのエレメントのリストは空となるであろう。

【2872】provider

新しい"TsTeleaddress" オブジェクトのプロバイダの値を形成するべき"TsOctets" オブジェクトへのポインタまたは、新しいテレアドレスオブジェクトがプロバオダの値を含まない場合には、0 ポインタ。

【2873】location

新しい"TsTeleaddress" オブジェクトのロケーションの値を形成するべき"TsString" オブジェクトへのポ

ンタかまたは、新しいテレアドレスオブジェクトがロケーションの値を含まない場合には、0 ポインタ。

#### 【2874】 Destruction

-TsTeleaddress:

-TsTeleaddress 0

” TsTeleaddress” オブジェクトを破壊する。” TsTeleaddress” オブジェクトおよびそのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。

#### 【2875】 Data Access

appendRoutingAdvice:

void appendRoutingAdvice (TsOctets

\* advice)

” TsOctets” オブジェクトに含まれるルーティングアドバイスのリストに、ルーティングアドバイスの単一のエレメントを、” TsOctets” オブジェクトの形で付加する。

#### 【2876】 advice

” TsTeleaddress” オブジェクトのルーティングアドバイスのリストに付加されるべきルーティングアドバイスのエレメントを伝える” TsTeleaddress” オブジェクトへのポインタ。

#### 【2877】 getProvider

const TsOctets \* getProvider() const

” TsTeleaddress” オブジェクトのプロバイダ値へのポインタをリターンするか、または、” TsTeleaddress” オブジェクトがプロバイダの値を持たない場合には、0 ポインタをリターンする。

#### 【2878】 getLocation:

const TString\*getLocation() const

” TsTeleaddress” オブジェクトのロケーションの値へのポインタをリターンするかまたは、” TsTeleaddress” オブジェクトがロケーションの値を持たない場合には0 ポインタをリターンする。

#### 【2879】 nextRoutingAdvice:

const TsOctets\*nextRoutingAdvice()

ルーティングアドバイスのエレメントへのポインタを” TsOctets” オブジェクトの形でリターンするかまたは0 ポインタをリターンする。” nextRoutingAdvice()” は多重の呼び出しを通じて、” TsTeleaddress” オブジェクトに含まれるルーティングアドバイスのエレメントのセットをそれらのエレメントが収容されていた順序でリターンする。

【2880】最後のルーティングアドバイスエレメントのリターンの後、リセットファンクソンが次に呼び出されるまで” nextRoutingAdvice()” 機能は、0 ポインタをリターンする。

#### 【2881】 reset:

void reset()

” TsTeleaddress” オブジェクトに含まれるルーティングアドバイスの最初のエレメントを、” nextRoutingAdo

vice()” のファンクションの次の呼び出しによってリターンする。

#### 【2882】 routingAdvice-Count:

size\_t routingAdviceCount() const

” TsTeleaddress” オブジェクトに現在含まれているルーティングアドバイスのエレメントの数をリターンする。

#### 【2883】 7. 17 Ts テレネーム (Ts Telename)

” TsTelename” クラスは、この開示のアペンディックス A に定義されたテレスクリプトクラス” TsTelename” の C++ 表現を供与する。” TsTelename” クラスのインプリメンテーションは、最小でも次の機能を供与しなければならない。

#### 【2884】 クラス TsTelename

{

public:

TsTelename (TsOctets \* authority,

TsOctets \* identity);

-TsTelename();

const TsOctets\*getAuthority() const; const TsOctets\*getIdentity() const;

Construction

TsTelename:

TsTelename (TsOctets\*Authority, TsOctets\*identity)

新しい” TsTelename” オブジェクトを構成する。

#### 【2885】 authority

新しい” TsTelename” オブジェクトの値を形成するべきオクテットストリングへのポインタ。

#### 【2886】 identity

新しい” TsTelename” オブジェクトのアイデンティティの値をオクテットストリングへのポインタまたは、” TsTelename” オブジェクトがアイデンティティの値を含まない場合、0 ポインタ。

#### 【2887】 Destruction

-TsTelename:

-TsTelename()

” TsTelename” オブジェクトを破壊する。” TsTelename” オブジェクトおよびこのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。

#### 【2888】 Data Access

getAuthority:

const TsOctets\*getAuthority() const

” TsTelename” オブジェクトに含まれるオーソリティの値へのポイントのリターンする。

#### 【2889】 getIdentity:

const TsOctets\*getIdentity() const

” TsTelename” オブジェクトに含まれるアイデンティティの値へのポイントのリターンするかまたは、” TsTelename” オブジェクトがアイデンティティの値を持たない

場合、0をリターンする。

【2890】7. 18 Tsテレナンバ (TsTele number)

"TsTelenumber" クラスは、この開示のアペンディックスAに定義されたテレスクリプトクラス" Telenumber" のC++表現を与える。" TsTelenumber" クラスのインプリメンテーションは最小でも次の機能を与えなければならない。

```
【2891】クラス TsTelenumber
{
public:
TsTelenumber (TsString * country,
TsString * telephone,
TsString * extension);
-TsTelenumber();
const TsString*getCountry() const;
const TsString*getTelephone() const;
const TsString * getExtension() const;};
Construction
TsTelenumber:
TsTelenumber(TsString*country, TsString*telephone,
TsString* extension)
```

新しい" TsTelenumber" オブジェクトを構成する。

【2892】country  
新しい" TsTelenumber" オブジェクトの国のコードの値を形成するべき" TsString" オブジェクトへのポインタか、または" TsTelenumber" オブジェクトが国のコードの値を含まない場合、0ポインタ。

【2893】注：供与された" TsString" オブジェクトに含まれる値は、CCITTが国または他の地理学的な領域に割り当てる数値コードのセットに制限される。

【2894】telephone  
新しい" TsTelenumber" オブジェクトのテレホンナンバの値を形成するべき" TsString" オブジェクトへのポインタまたは、" TsTelenumber" オブジェクトはテレホンナンバの値を含まない場合には、0ポインタ。 注：供与された" TsString" オブジェクトに含まれる値を形成するべきキャラクタは、数字 ("0" - "9")、ハイフン (" - ") およびスペースキャラクタ (" ") に制限される。

【2895】extension  
新しい" TsTelenumber" オブジェクトの内線番号の値を形成するべき" TsString" オブジェクトへのポインタまたは、" TsTelenumber" オブジェクトは内線番号の値を含まない場合には、0ポインタ。

【2896】注：供与された" TsString" オブジェクトに含まれる値を形成するキャラクタは、数字 ("0" - "9")、ハイフン (" - ") およびスペースキャラクタ (" ") に制限される。

【2897】Destruction

-TsTelenumber:

-TsTelenumber()

" TsTelenumber" オブジェクトを破壊する。" TsTelenumber" オブジェクトおよびこのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。

【2898】Data Access

getCountry:

```
const TsString * getCountry() const
新しい" TsTeleaddress" オブジェクトの国コードの値へのポインタをリターンするかまたは、" TsTeleaddress" オブジェクトが国のコードの値を含まない場合には、0ポインタをリターンする。
```

【2899】getIdentity:

```
const TsOctets*getIdentity() const
" TsTeleaddress" オブジェクトのテレホンナンバの値へのポインタをリターンするか、または、" TsTeleaddress" オブジェクトがテレホンナンバの値を含まない場合には、0ポインタをリターンする。
```

【2900】getExtension:

```
const TsString*getExtension() const
" TsTeleaddress" オブジェクトの内線番号の値へのポインタをリターンするか、または、" TsTeleaddress" オブジェクトが内線番号の値を含まない場合には0ポインタをリターンする。
```

【2901】7. 19 Tsウェイ (TsWay)

" TsWay" クラスは、この開示のアペンディックスAに定義されたテレスクリプトクラス" Way" のC++表現を供与する。" TsWay" クラスのインプリメンテーションは最小でも次の機能を与えなければならない。

【2902】クラス TsWay

```
{
public:
TsWay (TsTelename * name,
TsMeans * means,
TsObjSpecifier * secRegimeId);
-TsWay();
const TsTelename * getName() const; const TsMeans * getMeans() const; const TsObjSpecifier * getSecRegimeId() const;};
Construction
TsWay:
TsWay(TsTelename * name, TsMeans * means, TsObjSpecifier * secRegimeId)
新しい" TsWay" オブジェクトを構成する。
```

【2903】name

新しい" TsWay" オブジェクトのエンジン/領域テレネームの値を形成するべき" TsTelename" オブジェクトへのポインタ、または、新しい" TsWay" オブジェクトがエンジン/領域テレネームを含まない場合には、0ポインタ。

## 【2904】 means

新しい”TsWay”オブジェクトの平均値を形成するべき”TsMeans”オブジェクトへのポインタまたは、新しい”TsWay”オブジェクトが平均値を持たない場合には、0ポインタ。

## 【2905】 secRegimeId

新しい”TsWay”オブジェクトのセキュリティレジームの識別値を形成するべき”TsObjSpecifier”オブジェクトへのポインタ、または、新しい”TsWay”オブジェクトはセキュリティレジームの識別値を含まないが平均値を持たない場合には、0ポインタ。

## 【2906】 Destruction

-TsWay:

-TsWay 0

”TsWay”オブジェクトを破壊する。”TsWay”オブジェクトおよびそのオブジェクトがオーナーシップの責任を持つ全てのデータが破壊される。

## 【2907】 Data Access

getName:

const TsTelename \* getName 0 const

”TsWay”オブジェクトに含まれるエンジン/領域テレネームの値へのポインタをリターンするかまたは、”TsWay”オブジェクトはエンジン/領域テレネームを含まない場合には、0ポインタをリターンする。

## 【2908】 getMeans:

const TsMeans \* getMeans 0 const

Tsウェイオブジェクトに含まれる平均値へのポインタをリターンするか、または、”TsWay”オブジェクトは平均値を含まない場合には、0ポインタをリターンする。

## 【2909】 getSecRegimeId:

const TsObjSpecifier \* getSecRegimeId 0 const

”TsWay”オブジェクトに含まれるセキュリティレジームの識別子の値へのポインタをリターンするか、または、”TsWay”がセキュリティレジームの識別子の値を含まない場合には、0ポインタをリターンする。

## 【2910】 アベンディックスD

1. 序

1.1 パーミットの変更

1.2 他の変更

2 テレスクリプト概念

2.1 ブレイス間の伝送モデル

2.1.1 トラベルドブレイス

2.1.2 ショッピングボックス

2.1.3 オープンドオブジェクト

2.1.4 オープンドショッピングボックス

2.1.5 パーツボックス

2.1.6 チケットの使用

2.1.7 エンジンの役割

1. 序

このアベンディックスは、オブジェクトのインターチェンジおよびパーガトリイを含むように広く解釈される、ブレイスからブレイスへのトランスファ（移送）のメカニズムを記述する。

【2911】 このアベンディックスはこの開示のアベンディックスAと同様に構成されている。

【2912】 ブレイスからブレイスへの移送のメカニズムは、アベンディックスAの命令セットに対するいくつかの他の変更および付加を必要とし、これらは以下のサブセクションに記述される。

【2913】 1. 1 パーミットの変更 (Permit Changes)

”go”またはオペレーション”send”が成功しても失敗しても、オペレーション”entering”によってエージェントがリクエストしたものよりもより許容的な最初のローカルパーミットがエージェントに許容される。

【2914】 オペレーション”go”またはオペレーション”send”がフェイルしたとして、オペレーション”entering”によってエージェントを導入することができるとともに、エージェントには、エージェントがリクエストしたよりもより許容的でない最初のローカルパーミットが許容される。

【2915】 現在の領域がプロセスに供与するパーミットである、リードオンリパブリックインスタンス属性”regionalPermit”をクラス”Process”に付加する。

【2916】 プロセスのローカルパーミットに代わる新しい”changePermit”オペレーションによって、ブレイスの”ターミネート”オペレーションを変える。このオペレーションはブレイス自身または問題のプロセスの仲間の要求によって行なわれる。

【2917】 クラス”Process”にシステムインスタンスオペレーション”Permitchanged”を付加する。このオペレーションは、あるブレイスが占有者のローカルパーミットを変更する時にはいつでも、例えばオペレーション”entering”が上記のようにする際に、または猶予期間とともにもしくは猶予期間なしに占有者を排除するためにエンジンによって要求される。

【2918】 permitChanged:

op 0 PermitChanged|Nil;

レスポンドのローカルパーミットが例えばレスポンドが入るブレイスもしくはすでに占有しているブレイスによって変更された場合にいつでも、能力を付加または能力を除くために要求される。オペレーションの結果が0でなく例外であれば、エンジンはレスポンドの実行スレッドに例外を投出する。

【2919】 クラス”Process”に固有のこのオペレーションのための方法は単に0をリターンする。

【2920】 クラス”ExecutionException”にサブクラス”Permitchanged”を付加する。このサブクラスの投出数は、現在のプロセスのローカルパーミットが変更さ



れたことを意味する。

【2921】PermitChanged:

interface (ExecutionException)=0;

注：プロセスの正しいローカルパーミットが厳格に制限的である場合に、この例外をキャッチしてそれから回復するプロセスの能力は、プロセスの終了の規則によって制限される。

【2922】1. 2 他の変更 (Other Changes)

オペレーション” go” またはオペレーション” send” がフェイルした場合、プレイスからプレイスへの移送モデルに記述されているように、エージェントは、そのインタチェンジされた失われているオブジェクトの1個以上を見いだすことができる。

【2923】例えば、デスティネーションがエージェントにリクエストされたエージェントをパーミットすることを許容しないことによってオペレーションがフェイルしたとしても、” go” または” send” オペレーションによってエージェントをその目的点に到達させることができる。さらにオペレーション” entering” はこの場合にも存在している。

【2924】単にシステムフィーチャでなく予め定義されるかまたはユーザによって定義されたフィーチャをエンジンによってリクエストする。

【2925】エンジンによってリクエストされたフィーチャのための、ユーザによって定義された方法は、あたかもフィーチャがシステムフィーチャであるかのよう、クライアントおよびプロセスが0であることを見いだす。

【2926】現在の領域がプロセスの意図されたオーソリティにおいて持つ信用の程度をドキュメンテーションする、リードオンリパブリックインスタンス属性” authenticity” をクラス” Process” に付加する。この属性はアペンディックスAの4. 57に定義されている。

【2927】クラス” Process” は、プロセスが作り出された時点である、リードオンリパブリックインスタンス属性” creationTime” を付加する。

【2928】プロセスの属性” brand” を廃棄する。

【2929】注：その代わりある領域は、クラス” Permit” のサブクラスを定義し、その1つのインスタンスをプロセスの属性” regionalPermit” として供給することができる。

【2930】2 テレスク립ト概念 (TELESCRIPT CONCEPTS)

2. 1 プレイス間の伝送モデル (Place-to-place Transfer Model)

命令セットは、このセクションが定義する” プレイスからプレイスへの移行モデル (place-to-place transfer model)” を実現する。

【2931】2. 1. 1 トラベルドプレイス (Travel

veled Places)

” Traveled” プレイスは、 SHIPPINGボックスがそれを通して移送されるプレイスである。

【2932】Transfer

SHIPPINGボックスの” origin” である1つのプレイスからSHIPPINGボックスの” destination” である1以上のプレイスにボックスの内容物を” transfer” するようにネットワークがあるSHIPPINGボックスのコーポレートに委託した、トラブルされるプレイス (複数)。トラブルされるプレイスは、異なった方向にSHIPPINGボックスの内容物を搬送しなければならない時にはいつでも、新しいSHIPPINGボックスを作り出す。

【2933】SHIPPINGボックスの内容物をそのオリジンからそのデスティネーションのどれか1つにトランスファするのは、4つのステップで行なわれ、その最後の2つすなわちトランスファインおよびトランスファアウトは、単一のトランスファ内において反復することができる。

【2934】Submission

トランスファはSHIPPINGボックスの” submission” によって開始される。SHIPPINGボックスが結果として最初にトランスファインされるトラブルされるプレイスはその” source” である。

【2935】Delivery

トランスファは、SHIPPINGボックスの” delivery” で終了する。サブミットされたSHIPPINGボックスには配達される場所がその” destination” である。

【2936】SHIPPINGボックスの配送は” normal” または” abnormal” である。前者の場合SHIPPINGボックスの目的点は、サブミッションにおいて特定された通りであり、SHIPPINGボックスの内容物の全ての臨界の部分が存在している。第2の場合には、ネットワークは、SHIPPINGボックスのための別途の予め規定されたデスティネーションを持っており、SHIPPINGボックスの内容物のある臨界な部分が存在していないことがある。例外はこの異常性を説明する。

【2937】Transfer In

SHIPPINGボックスは、トラブルされたプレイスに” transferred in” される。トランスファインは、あるプレイス (トラブルされるプレイスでも他の場所でもよい) からのSHIPPINGボックスのサブミッションの結果であるか、または、トラブルされたプレイス (SHIPPINGボックスがトランスファインされるトラブルされる場所と同一でも異なってもよい) からのSHIPPINGボックスのトランスファアウトの結果である。

【2938】SHIPPINGボックスがそれから来たプレイスは、SHIPPINGボックスの” previous hop” である。

【2939】Transfer Out

SHIPPINGボックスは、トラブルされるプレイスから” transferred out” される。トランスファアウトは、ト

ラブルされるブレイス（ SHIPPINGボックスがそれからトランスファアウトされたトラブルされる場所と同一でも異なっているとしてもよい）への SHIPPINGボックスのトランスファインの結果であるか、ブレイス（トラブルされるブレイスでもその他のブレイスでもよい）への SHIPPINGボックスの配送の結果である。

【2940】 SHIPPINGボックスがそれに向かう場所（単数もしくは複数）は SHIPPINGボックスの "next hop" である。

【2941】 2. 1. 2 ショッピングボックス (Shipping Boxes)  
"Shipping Boxes" は、別のものすなわちその "contents" を保持し、そのオブジェクトをブレイスからブレイスへと移送させるために使用することができる1つのオブジェクトである。

【2942】 Peek  
トラブルされるブレイスは、 SHIPPINGボックスの内容物であるオブジェクトに "peek" することができる。すなわち、トラブルされるブレイスは、下記の制限に従って、オブジェクトの属性を取得することができる。

【2943】 SHIPPINGボックスは、 SHIPPINGボックスの内容物へのプロテクトされるリファレンスを用いて SHIPPINGボックスの内容物の属性を取得する。ユーザによって定義された方法が関与していれば、方法は、あたかも属性がシステムの属性であるかのように、 "client"、 "here" および "Process" が0であることを見いだす。またある属性をピークすることは、その予め定められた通路を無視し、どの場合にも、属性のオリジナルでなく属性のコピーをリターンする。

【2944】 Poke  
トラブルされるブレイスは、 SHIPPINGボックスの内容物であるオブジェクトに "poke" することができる。すなわち、トラブルされるブレイスは、下記の制限に従って、オブジェクトの属性をセットすることができる。しかし、トラブルされるブレイスは、トラブルされるブレイスが適当に特権を持っている場合にのみこれを行ないうる。

【2945】 SHIPPINGボックスは、 SHIPPINGボックスの内容物への保護されないリファレンスを用いて、その内容物の属性をセットする。ユーザによって定義された方法が関与していれば、その方法は、あたかも属性がシステムの属性であるかのように、方法は、 "client"、 "here" および "Process" が0であることを見いだす。さらに、属性においてポーキングすると、どんな場合にも、提案された属性のオリジナルでなくそのコピーを用いて、予め定められた通路が無視される。

【2946】 前記の特権は次のようにして割り当てられる。あるエンジンブレイスは、たとえ命令セットがリードオンリとして定義していても、属性 "authentication State" または "regionalPermit" をセットすることが

できる。 "regionalPermit" 属性の先取的なセッティングは、 SHIPPINGボックスの異常な配送を規定する。

【2947】 注：トラブルされるブレイスは、 SHIPPINGボックスの内容物のオペレーションをリクエストすることはできない。

【2948】 Rebox

トラブルされるブレイスは、 SHIPPINGボックスを "rebox" することができる。トラブルされるブレイスは、 SHIPPINGボックスは異なる方向に移送されねばならない場合にそれを行なう。エンジンは、 SHIPPINGボックスから特定の目的点のチケットを除去し、それらを0によって代替し、現存のものと内容が同一の新しい SHIPPINGボックスにそれを付け替える。

【2949】 Redirect

トラブルされるブレイスは、 SHIPPINGボックスを "redirect" することができる。トラブルされるブレイスは、そのトランスファの間に例外が生じた場合にこれを行なう。エンジンは、 SHIPPINGボックスへの例外を付着させ、 SHIPPINGボックスの目的点チケットの中から0を除去し、単一の特定されるチケットを各々の残存する目的点チケットに変えることができる。

【2950】 2. 1. 3 オープンドオブジェクト (Opened Objects)

"Opened" オブジェクトは、その0以上の部分がマニピュレート可能なオブジェクトである。

【2951】 Parts

オブジェクトの各々の "part" は開放されていてもいなくても、そのダイジェストが0でないインターチェンジドオブジェクトである。開放されるオブジェクトの一部分は、そのパートのクラスおよびダイジェストの別のオブジェクトに、開放されるオブジェクトに大きく影響することなく代替することができる。

【2952】 Presence

開放されていてもいなくてもあるオブジェクトの各々の部分は、 "present" または不在である。一般にある部分の "absence" は、オブジェクトの使用を制限するが、オブジェクトのブレイスからブレイスへの移送を一層効率的にする。あるオブジェクトはその時点において存在しないそれ自身の一部分へのリファレンスを使用しようと試みた場合、エンジンは "Reference Void" を投出する。

【2953】 注：典型的なプラクティスにおいては、 SHIPPINGボックスの内容物の一部は SHIPPINGボックスの委託点（サブミッション）において存在し再び SHIPPINGボックスの配送点において存在するが、作用を最適にするために、 SHIPPINGボックスのトランスファの他のステップの全部または一部では存在しない。

【2954】 Criticality

開放されていてもいなくても、オブジェクトの各々の部分は、臨界であるかまたは非臨界である。オブジェクト

は、オブジェクトの"non-critical"部分の1以上が存在していなくても使用されるように設計されているが、オブジェクトの"critical"なパートが、どれか存在していない場合にはそうではない。

【2955】注：あるオブジェクトの臨界の部分の不在は、そのオブジェクトを収容した SHIPPING BOXES の正常な配送を妨げるが、非臨界な部分の不在はそうではない。

【2956】注：典型的なプラクティスにおいて、オブジェクトの全ての部分はあらゆる場所で臨界である。時にはあるオブジェクトのある部分は、そのオブジェクトが臨界なプレイスにいる間のみ臨界である。

【2957】2. 1. 4 オープンド SHIPPING BOXES (Opened Shipping Boxes)  
"Opened Shipping Boxes" は、そのパーツが操作されるボックスである。開放される SHIPPING BOXES の部分は SHIPPING BOXES の内容物の部分である。

【2958】2. 1. 5 パーツボックス (Parts Boxes)

"Parts Boxes" は、その部分が明文でかまたは任意に特定することができる開放されたオブジェクトである。パーツボックスは次の4つの目的に用いられる。

#### 【2959】Manipulating Parts

開放されたオブジェクトの部分はパーツボックスによってマニピュレートされる。これによってマニピュレーションを行なうオブジェクトにパーツが直接近接可能でないことが保証される。例えばパーツボックス中のパーツは、開放されたオブジェクトに含めることができ、開放されたオブジェクトの指定された部分は、コピーされるかまたはパーツボックスに移すことができる。パーツ自身はパーツボックス中にある間も検査できない。

【2960】注：これによってインターチェンジされたオブジェクトによって表される知的所有権が保護される。

#### 【2961】Storing Parts

開放された SHIPPING BOXES が委託されたトラブルされるプレイスは、これらの SHIPPING BOXES に含まれる部品のための貯蔵所としてパーツボックスを使用することができる。パーツボックスは小さなデータボックスであり、プレイスからプレイスへのトランスファにおいてリソースとして役立つ。

【2962】注：これによってトラブルされるプレイスのインプリメンテーションが簡単になる。

#### 【2963】Taking Parts

SHIPPING BOXES の内容物がパーツボックスを含む場合、パーツボックスの臨界の部分が特定の SHIPPING BOXES のパーツボックスから除かれる。そのため SHIPPING BOXES に関与するトラブルされるプレイスはこれらのパーツを調べることも除外することもできない。このような除外が表す最適化は試みられない。

【2964】注：これによって、プレイスからプレイスへのトランスファを支持するエージェントは、インターチェンジされるオブジェクトをトラベルされるプレイスの間に移動させることができ、システム全体がシステムの（この場合は誤った方向の）最適化の努力によってこの運動を避けようと試みることはない。

#### 【2965】Leaving Parts

SHIPPING BOXES の内容物がパーツボックスを含む場合、パーツボックスの非臨界の部分は特異的に、SHIPPING BOXES に含められる。このようにしてあるエージェントは、インターチェンジされたオブジェクトが非臨界であることを宣言することができ、その後この部分が必然的にエージェントに不随することなく、トリップを行なうことができる。後にパーツボックスからパートを取り除くことによって、エージェントは、そのパートが臨界であることを宣言する。その場合、パーツが不在ならば、エンジンはそのパーツをローカルに見い出さず、またはエンジンは、そのパーツがエージェントによって使用されている場合、"Reference Void" を投出する。

【2966】注：エージェントはこれによって、そのインターチェンジされたオブジェクトのあるものがそこに存在することを主張することなく、それらのオブジェクトのあるものが使用される場所を訪問することができる。

#### 【2967】2. 1. 6 チケットの使用 (The Use of Tickets)

チケットは後述するように、SHIPPING BOXES がそれを通して移送される複数のプレイスを規定する。"place attributes" はこのセクション全体を通じて、"destinationName"、"destinationAddress" および "destinationClass" の各属性である。以下に特に説明しない他のチケット特性は無視される。

#### 【2968】Source

チケットは SHIPPING BOXES のソースを規定するために用いられる。チケットの場所の属性（ソースを特定する）は、割り当てられたテレネーム、割り当てられたアドレスおよび割り当てられたサイテーションである。チケットの"way"の属性は、0 でなければ、ソースが必要に応じて到達するべきウェイを指定する。

#### 【2969】Destination

チケットは、SHIPPING BOXES の各々の目的点を規定するために用いられる。チケットの場所の属性は、問題の目的点を特定する。チケットの属性"way"は、0 でなければ、その目的点が到達するべきウェイを指定する。"desiredWait" および "maximumWait" の各属性は、SHIPPING BOXES がその目的点に向かって移送されている間 SHIPPING BOXES に供与されるべきサービスの品質を与える。

#### 【2970】Previous Hop

チケットは、 SHIPPINGボックスのプリビースホップを規定するために用いられる。チケットの属性"destinationName"、"destinationAddress" および"destinationClass" (プリビースホップを特定する) はそれぞれ割り当てられたテレネーム、割り当てられたテレアドレスおよび割り当てられたサイテーションである。チケットの属性"way" は0でなければ、SHIPPINGボックスがプリビースホップからトランスファされたウェイを規定する。

#### 【2971】Next Hop

チケットは、SHIPPINGボックスのネクストホップを規定するために用いられる。チケットの属性プレイスはネクストホップを特定する。チケットの属性"way" は0でなければ、ネクストホップが到達されるべきウェイを指定する。

#### 【2972】2. 1. 7 エンジンの役割 (The Engine's Role)

トラブルされるプレイスは、SHIPPINGボックスの移送とその目的点への配送とを、制御するのでなく、容易にするものとして見るのが適切である。エンジン自身が次のようにこのプロセスにおいて手助けをする。

#### 【2973】Transfer In

トラブルされるプレイスはSHIPPINGボックスにおいてトランスファされる時はいつでもエンジンはそのSHIPPINGボックスが開放されたものかそれともそうでないかをすでに定めている。典型的なプラクティスではエンジンはボックスごとにでなく統一的にこの決定を行なう。従って一般に、あるトラブルされるプレイスは、SHIPPINGボックスの内容物の一部分の相互変換を付託されているが他のものはそうではない。

#### 【2974】Transfer Out

トラブルされるプレイスがSHIPPINGボックス" A" をトランスファアウトする時はいつでもエンジンは次のことを行なう。第1にエンジンは" A" をリボックスし、新しいSHIPPINGボックス" B" に移行し、またエンジンがSHIPPINGボックスを配送することができるの目的点へもチケットを移動させる。エンジンはトランスファアウトが配送を許容するものである場合にのみチケットを" B" に移動させる。第2にエンジンは、" A" を再びリボックスし、全ての残存するチケットを別の新しいSHIPPINGボックス" C" に移動させる。第3にエンジンは、非同期的に" C" をトランスファし、" B" を配送することを試みる。

#### 【2975】Expiration

トラブルされるプレイスがSHIPPINGボックス" A" のどれかの目的点チケットを消滅させることを可能とする時、エンジンは次のことを行なう。このようなチケットは、その最大期間が到達した時に" 消滅 (expires)" する。エンジンは、第1に、" A" をリボックスし、消滅したチケットを新しいSHIPPINGボックス" B" に移

動させる。第2にエンジンは、" B" を、エンジンによって選択されたプレイスに再指向させ、異常の結果として、" Ticket Expired" をレコードする。第3にエンジンは、" B" を、エンジンによって選択されたプレイスにトランスファアウトし、これは、トラブルされたプレイスがそれを行なったかのように行なわれる。

#### 【2976】Discard

デスティネーションチケットがそれに対し固定されているSHIPPINGボックス" A" の最終化を惹起させるように、このSHIPPINGボックス" A" をあるトラブルされるプレイスが無視する時はいつでも、エンジンが次のことを行なう。第1にエンジンは、" A" をリボックスし、全てのデスティネーションチケットを新しいSHIPPINGボックス" B" に移動させる。第2にエンジンは" A" を破壊する。第3にエンジンは、トラブルされるプレイスがそれを行なったかのように、エンジン自身が選択したプレイスに、" B" を非同期的にトランスファアウトする。

#### 【2977】2. 1. 8 ルーティングにおけるエンジンの役割

エンジンは、SHIPPINGボックスの第2ホップを選択し、SHIPPINGホップの第2ホップから最後のホップまでを拘束し、さらにSHIPPINGボックスの最初のホップを、ある領域内のホップの任意の順序において制限することによって、SHIPPINGボックスがそれにそって移送されるルート制限する。

#### 【2978】First Regional Hop

ある領域のエンジンは、その領域に入るSHIPPINGボックスが、その領域内の別のトラベルされるプレイスにトランスファインされたりその領域内において移送されたりする前に、エンジンプレイスにトランスファインされることを保証するように互いに共同する。

#### 【2979】Second Hop

エンジンは、SHIPPINGボックスのソースから、SHIPPINGボックスのセカンドホップすなわちSHIPPINGボックスが最初にトランスファインされるトラブルされるプレイスを次のようにして選択する。

【2980】第2のホップは、SHIPPINGボックスのサブミッションボックス (もしそれがトラブルされるプレイスであれば) であるかまたはさもなければサブミッションプレイスのスーパープレイスである。いくつかのスーパープレイスがトラブルされるプレイスである場合他のものを全く含まないものが選択される。

【2981】注：従って" submission" は、SHIPPINGボックスがトラブルされるプレイスに到達するまでSHIPPINGボックスがそのオリジン及び次々のスーパープレイスを通してフォール (fall) させるものとして理解することができる。

#### 【2982】Second-to-last Hop

エンジンは次のようにして、SHIPPINGボックスの終わ

りから2番目のホップすなわち SHIPPING ボックスが最後にトランスファアウトされたトラブルされるプレイスを、SHIPPING ボックスの全ての目的点に対してコンストレイントする。

【2983】終わってから2番目のホップは、目的点それ自身であるか、そのスーパープレイスの1つである。これはSHIPPING ボックスの目的点の各々について個別にそうなる。

【2984】注：従って“delivery”は、SHIPPING ボックスがその各々の目的点として資格を持つものに到達するまで連続するサブプレイスを通してSHIPPING ボックスをライズ（rise）させるものと理解することができる。

【2985】最後から2番目のホップが与えられた場合、もしいくつかのプレイスがSHIPPING ボックスの目的点の1つとして資格を持つならば、エンジンはSHIPPING ボックスをそれらの内の1つに配送しようと試み、アペンディックスAのセクション2. 5. 4に示した順序でそれらに接近する。

【2986】最後から2番目のホップが与えられたとして、どのプレイスもSHIPPING ボックスの目的点の1つとして資格がなくまたはどのプレイスもSHIPPING ボックスの引き渡しを受け付けられない場合には、エンジンはSHIPPING ボックスをエンジン自身が選んだ1つのプレイスにトランスファアウトする。

【2987】Non-delivery

エンジンはあるSHIPPING ボックスが最終的に配送不可能であることをシステムエラーとして処理しこのシステムエラーをエンジンは現行のOAMポリシーに従って取り扱う。

【2988】注：エンジンは、他の仕方では引き渡しのできないSHIPPING ボックスのために最後によるプレイスとして“purgatory”を供給することができる。このプレイスは占有者のローカルパーミットを著しく制限することができる。

【2989】2. 1. 9 ゴーの実行（Implementing Go）

クラス“agent”に固有の“go”の方法は、エージェント及びエージェントが所有するオブジェクトを内容とするSHIPPING ボックスを用いて次のようにオペレーションをインプリメントする。

【2990】Submission

エンジンは、オペレーションについてのエージェントのリクエストを肯定した後、エージェントを含むSHIPPING ボックスを作り出しそしてサブミット（submit）する。従ってSHIPPING ボックスの最初のホップすなわちそのオリジンは、エージェントがオペレーションを要求した時にエージェントが占有していたプレイスである。

【2991】Delivery

エンジンはオペレーションを終了することによってシッ

ピングボックスを配送する。従ってSHIPPING ホップのラストホップすなわちその唯一つの目的点は、エージェントがそのオペレーションを完了した時にエージェントが占有しているプレイスである。

【2992】オペレーションは、SHIPPING ボックスが例外を記述せずSHIPPING ボックスの非臨界的な部分のみが不在な時に成功しその他の場合には失敗する。オペレーションが失敗すれば、オペレーションは、SHIPPING ボックスが記録する例外を投出（SHIPPING ボックスがその例外を記録していれば）しさもなければエンジン自身が選んだ例外を投出する。

【2993】2. 1. 10 センドの実行（Implementing Send）

クラス“agent”に固有のオペレーション“send”の方法は、オペレーション“go”がインプリメントされるのと同じ仕方では“send”オペレーションをインプリメントするが2つの違いがある。第1にSHIPPING ボックスは、移送すべきクローンと同じ数の目的点を持つ。第2にSHIPPING ボックスの内容物は、オペレーションをリクエストしたエージェントの代表的なクローンである。クロンの割り当てられたテレネームの属性“identity”は、委託の時には0であるが各々のクロンが配送されるにつれて異なった仕方ではセットされる。

【2994】3 テレسكريプトクラスの外觀

3. 1 プレイス間の伝送グループ

オブジェクト（参照される）

- ・ Parts Box（開放）
- ・ Shipping Box（移動しない）
- ・ Opened Shipping Box（開放）

Opened

Traveled

3. 1. 1 開放（Opened）

オペレーション

clearParts

examineParts

excludeParts

getClasses

getDigests

getNumber

includeParts

“opened”オブジェクトは、そのパーツが操作可能なあるオブジェクトである。開放されたオブジェクトの固有のオペレーションは、指定された基準を充たすパーツのナンバ（オペレーション“getNumber”）、クラス（オペレーション“getClasses”）、及びダイジェスト（オペレーション“getDigests”）を明らかにし、新しいパーツを含め（オペレーション“includeParts”）、現存するパーツを排除し（オペレーション“excludeParts”）特定の基準を充たす全ての現存のパーツを排除し（オペレーション“clearParts”）そして特定のパーツ

の検査の準備をする（オペレーション” examineParts”）。

【2995】3. 1. 2 開放された SHIPPING ボックス

” opened shipping box” はそのパーツがマニピュレート可能な SHIPPING ボックスである。

【2996】3. 1. 3 パーツボックス

オペレーション

includeParts

” parts box” はそのパーツが指定されるとともにマニピュレートもすることができるオブジェクトである。

【2997】パーツボックスの固有のオペレーションは新しいパーツを含む（オペレーション” includeParts”）。

【2998】3. 1. 4 SHIPPING ボックス

Attributes

destinations

exception

source

time

オペレーション

peek

poke

rebox

redirect

” shipping box” は、ブレイスの間をトランスファすることができるオブジェクトである。

【2999】SHIPPING ボックスの固有の属性は、SHIPPING ボックスが委託された時間（属性” time”）とSHIPPING ボックスのソースへのチケット（属性” source”）と、SHIPPING ボックスの目的点へのチケット（属性” destinations”）と、SHIPPING ボックスの移送の間に遭遇することのある例外（属性” exception”）である。

【3000】SHIPPING ボックスの固有のオペレーションは、SHIPPING ボックスの内容物の特定された属性を取得（オペレーション” peek”）または設定（オペレーション” poke”）し、例外の場合にSHIPPING ボックスを再指向（オペレーション” redirect”）させ、いくつかのネクストホップを許容するようにSHIPPING ボックスをレボックスする（オペレーション” rebox”）。

【3001】3. 1. 5 トラベルド (Traveled)

オペレーション

transferOut

transferredIn

” traveled” オブジェクトは、SHIPPING ボックスを移送させる。

【3002】トラブルされたブレイスの固有のオペレーションは、SHIPPING ボックスをトランスファアウトし（オペレーション” transferOut”）SHIPPING ボック

スのトランスファインに作用する（オペレーション” transferredIn”）。

【3003】4 テレスクリプトクラスの詳細

4. 1 開放 (Opened)

Opened

Class

Opened:

シールされたアブストラクトなインタフェース 0 = (..);

パブリックインスタンスオペレーション

clearParts:

プロテクトされない op (isCritical: Boolean|Nil);

ブーリアン（アークギュメント” isCritical”）が規定する規格を満たすレスポンドの現在のパーツの全てをレスポンドから除去して廃棄する。下記オペレーション” getClasses” 参照。

【3004】examineParts:

op ( ofClass: Class;

digests: protected Set [Object])

PartsBox|Nil;

そのクラス（アークギュメント” ofClass”）及びダイジェスト（アークギュメント” digests”）が指定されている現在のパーツをレスポンドに残し、リターンする。オペレーションはパーツを含むパーツボックス（もしあれば）をリターンするかさもなければ0をリターンする。

【3005】excludeParts:

プロテクトされない op (ofClass: Class;

digests: protected Set [Object])

PastsBox|Nil;

そのクラス（アークギュメント” ofClass”）及びダイジェスト（アークギュメント” digests”）が指定されている現在のパーツをレスポンドから除去しリターンする。オペレーションはパーツ（もしあれば）を含むパーツボックスをリターンするかさもなければ0をリターンする。

【3006】getClasses:

op (isPresent, isritical: Boolean|Nil)

protected Set [Class];

ブーリアン（アークギュメント” isPresent”及び” isCritical”）が規定する基準（後述する）を満たすレスポンドのパーツのクラスをリターンする。

【3007】第1のブーリアンによって規定される第1の基準は、ブーリアンが真であればパーツは存在し、ブーリアンが偽であればパーツは存在せず、アークギュメントが0であればその両方であるということである。第2のブーリアンによって規定される第2の基準は、ブーリアンが真であればパーツは臨界であり、ブーリアンが偽であればパーツは非臨界であり、アークギュメントが0であればその両方である、ということである。

【3008】getDigests:

op(ofClass: Class; isPresent, isCritical: Boolean|Nil)

protected Set [Object]

指定されたクラス（アーギュメント” ofClass”）であり、ブーリアン（アーギュメント” isPresent” 及び” isCritical”）が規定する基準（前記オペレーション” getClasses” 参照）を充たすレスポンドのパーツのダイジェストをリターンする。

【3009】 getNumber:

op(isPresent, isCritical: Boolean|Nil) Integer;  
ブーリアン（” isPresent” 及び” isCritical”）が規定するクリテリヤ（前記オペレーション” getClasses” 参照）を充たすレスポンドのパーツの番号をリターンする。

【3010】 includeParts:

プロテクトされない op (parts:protected PartsBox);  
レスポンドのパーツにクラス及びダイジェストが適合するパーツボックスのパーツ（アーギュメント” parts”）をレスポンドに含める。オペレーションは、最初に、そこにすでに存在している適合したパーツをレスポンドから排除し、除去する。

【3011】 4. 2 開放された SHIPPING ボックス  
オブジェクト（参照される）

- ・ SHIPPING ボックス（移動しない）
- ・ Opened Shipping Box（開放）

Class

OpenedShippingBox:

シールドインタフェース (ShippingBox, Opened) = 0;

4. 3 パーツボックス

オブジェクト（参照される）

- ・ Parts Box（開放）

Class

PartsBox:

シールドインタフェース（オブジェクト、開放される）  
= (...);

Construction

initialize:

プロテクトされない op (

criticalParts, noncriticalParts: Set  
[Interchanged] |Nil);

相互変換されたオブジェクトの2つのセットを、レスポンドの臨界の（アーギュメント” criticalParts”）パーツ及び非臨界の（アーギュメント” non-critical”）パーツ（全て最初に存在している）とする。しかしセットの代わりに0が存在していれば、セットはクリアされたものと推定する。

【3012】 Public Instance Operations

” includeParts”:

プロテクトされない op (parts:protected PartsBox);  
パーツボックスのパーツ（アーギュメント” parts”）

をレスポンドに含める。各々のパーツはレスポンドに存在していると思われる。各々のパーツは、それがパーツボックスにおいてそのように考えられている場合にのみレスポンド中において臨界であると思われる。オペレーションは最初に、すでにそこに存在している適合するパーツをレスポンドから排除しそして廃棄する。

【3013】 4. 4 SHIPPING ボックス

オブジェクト（参照される）

- ・ Shipping Box（移動しない）

Class

ShippingBox:

シールドインタフェース（オブジェクト、移動しない）  
= (...);

Construction

initialize

プロテクトされない op 0

FeatureUnavailableを送出;

例外を送出(”FeatureUnavailable”).

【3014】 注: SHIPPING ボックスはオペレーション” new” を使用することなくオペレーション” go” または” send” によって作り出される。

【3015】 Public Instance Attributes

destination:

リードオンリのプロテクトされる List [Ticket|Nil];  
レスポンドの目的点及び0またはそれ以上の0（オペレーション” rebox” を用いて他のSHIPPING ボックスに移動したチケットに変えられる）を規定するチケット。

【3016】 exception

リードオンリ TripException|Nil;

実際にトリップの例外に遭遇した場合に、レスポンドを移送する間に遭遇したトリップの例外さもなければ0。

【3017】 source

リードオンリのプロテクトされるチケット。

【3018】 レスポンドのソースを規定するチケット。

【3019】 time:

リードオンリのタイム。

【3020】 レスポンドが委託された時間。

【3021】 パブリックインスタンスオペレーション

peek:

op(identifier: Identifier!) copiedObject

CopyUnavailable, FeatureUnavailableを送出する;

識別子（アーギュメント” identifier”）が表すレスポンドの内容物の属性をリターンする。

【3022】 リクエストにとって近接可能なレスポンドの内容の属性を識別子が表さない場合（” FeatureUnavailable”）または属性のコピーが利用できない場合（” CopyUnavailable”）例外が投出される。

【3023】 poke:

プロテクトされない op (identifier:Identifier!; at  
tribute: copiedObject)

ArgumentInvalid, AttributeReadOnly, FeatureUnavailableを送出する;

オブジェクト(アークギュメント"attribute")を識別子(アークギュメント"identifier")が表すレスポンドの内容物の属性にする。オペレーションは最初に属性(もし存在すれば)を廃棄する。

【3024】オブジェクトは属性のコンストレイントに違反しているか("ArgumentInvalid")、属性がリードオンリであるか("AttributeReadOnly")または識別子がリクエストにとって近接可能なレスポンドの内容物の属性を表さない場合("FeatureUnavailable")例外が投出される。

【3025】rebox

プロテクトされない op (チケット:protected Set[Integer]) ShippingBox

throws PositionInvalid;

その内容がレスポンドのものと同一である新しい SHIPPINGボックスを作り出してリターンし、レスポンドの属性"destinations"中の指定された位置にあるチケット("tickets")を新しいSHIPPINGボックスに移動し、それらを0で代替する。

【3026】指定された位置がそのようなものとして不適切であれば("PositionInvalid")例外が投出される。

【3027】redirect:

プロテクトされない op (例外:TripException;

newDestination: copied Ticket|Nil);

レスポンドの"destinations"属性から各々のニルアイテムを除去し、残っている各々のアイテムを、もしあればチケット(アークギュメント"newDestination")で置き換え、例外(アークギュメント"exception")をレスポンドの属性"exception"にする。

【3028】アダプテーション

copy

Throws Copy Unavailable.

finalize

モデルによって規定されたようにレスポンドをレボックスする。

【3029】4. 5   トラベルド (Traveled)

Traveled

Class

Traveled:

アブストラクトインタフェース 0 = (...);

プライベートインスタンスオペレーション

transferOut:

プロテクトされない op (shippingBox: unprotected shippingBox;

suggestedNextHop: protected

Ticket|Nil;

canDeliver: Boolean);

0でなければ、チケット(アークギュメント"suggestedNextHop")によって規定されるプロセスに、さもなければエンジンによって選択されるプレイスに、SHIPPINGボックス(アークギュメント"shippingBox")をトランスファアウトする。たとえチケットが存在していても、チケットは、次のホップがチケットによって規定されるプレイスであることを要求するのではなく示唆する。

【3030】ブーリアン(アークギュメント"canDeliver")、エンジンがそのようにすることができるならばエンジンがSHIPPINGボックスを自由に配送することができるかを指示する。

【3031】システムインスタンスオペレーション transferredIn:

アブストラクトのプロテクトされていない op (shippingBox: unprotectedShippingBox;

previousHop: protected Ticket);

SHIPPINGボックス(アークギュメント"ShippingBox")がチケット(アークギュメント"previousHop")によって規定されたプレイスからトランスファインされる時にリクエストされる。

【3032】マイクロフィルムアペンディックスGのコンピュータプログラムは、カリフォルニア、マウンテンビュー、シリコングラフィックスから入手されるアイリスインディゴコンピュータシステムのようなワークステーションを備えた、ユニックスオペレーティングシステムIRIX4. 0. 5、コンパイラ及びリンクを用いて、1次実施例に従ってコンパイルされリンクされた。第2実施例によれば、マイクロアペンディックスGのコンピュータプログラムは、MPW C++コンパイラを用いてコンパイルされ、MPWリンクを用いてリンクされた。これら両者は、カリフォルニア州クーパーチノ、アップルコンピュータINC. から入手され、やはりアップルコンピュータINC.から入手されるアップルマッキントッシュ(登録商標)コンピュータにおいて使用することができる。使用される特別なコンピュータ言語及びコンピュータシステムは本発明の重要な要素ではない。この開示に基づいて、同業者は、いろいろのコンピュータ言語及び/又はいろいろのコンピュータシステムを用いて本発明を実施することができる。

【図面の簡単な説明】

【図1】従来技術によるコンピュータシステムの構造を示す図である。。

【図2】リモートプロシージャコーリングを使用した従来技術による方法のフローチャートである。

【図3】リモートプロシージャコーリングを具体化する従来技術によるネットワークを示す図である。

【図4】リモートプログラミングを使用する従来技術による方法のフローチャートである。

【図5】リモートプログラミングを具体化する従来技術によるネットワークを示す図である。



【図6】本発明に従って構成されたネットワークを通るエージェントプロセスの運動を示す図である。

【図7】本発明に従って構成されたネットワークを通るエージェントプロセスの運動を示す図である。

【図8】本発明の原理に従って形成されたコンピュータシステムにおいて実行されるプロセス（複数）を示す図である。

【図9】本発明の原理に従って形成されたコンピュータシステムにおいて実行されるプロセス（複数）を示す図である。

【図10】本発明の一実施例によるクラス階層の一部を示す図である。

【図11】本発明に従って形成されたネットワークを通るエージェントプロセスの運動を示す図である。

【図12】本発明に従って形成されたネットワークを通るエージェントプロセスの運動を示す図である。

【図13】本発明に従って形成されたネットワークを通るエージェントプロセスの運動を示す図である。

【図14】本発明の一局面に従って形成されたネットワークを通るエージェントプロセスのクローン（複数）の運動を示す図である。

【図15】本発明の一局面に従って形成されたネットワークを通るエージェントプロセスのクローン（複数）の運動を示す図である。

【図16】本発明の一局面に従って形成されたネットワークを通るエージェントプロセスのクローン（複数）の運動を示す図である。

【図17】本発明の一局面に従って形成されたネットワークを通るエージェントプロセスのクローン（複数）の運動を示す図である。

【図18】本発明の一局面に従って形成されたネットワークを通るエージェントプロセスのクローン（複数）の運動を示す図である。

【図19】オペレーション”meet”を使用することによる2つのエージェントプロセスの間の相互作用を示す図である。

【図20】オペレーション”meet”のオペレーションを使用することによる2つのエージェントプロセスの間の相互作用を示す図である。

【図21】オペレーション”meet”を使用することによる2つのエージェントプロセスの間の相互作用を示す図である。

【図22】オペレーション”meet”を使用することによる2つのエージェントプロセスの間の相互作用を示す図である。

【図23】オペレーション”meet”を使用することによる2つのエージェントプロセスの間の相互作用を示す図である。

【図24】オペレーション”meet”を使用することによる2つのエージェントプロセスの間の相互作用を示す図

である。

【図25】本発明の一局面に従って形成されたブレイスプロセスの相互作用を示す図である。

【図26】本発明の一局面に従って形成されたブレイスプロセスの相互作用を示す図である。

【図27】本発明の一局面に従って形成されたブレイスプロセスの相互作用を示す図である。

【図28】本発明の一局面に従って形成されたブレイスプロセスの相互作用を示す図である。

【図29】本発明の一局面に従って形成されたブレイスプロセスの相互作用を示す図である。

【図30】本発明に従って構成されたコンピュータネットワークの構成を示す図である。

【図31】図30に示したネットワークの別の表現形態を現す図である。

【図32】図30に示したネットワークの別の表現形態を現す図である。

【図33】本発明の一実施例によるエージェントプロセスのクラス関係を示す線図である。

【図34】本発明に従って形成されたネットワークの構造を示すと共に、ネットワークを通るあるエージェントプロセスの運動を示す図である。

【図35】オペレーション”go”の遂行による本発明のコンピュータネットワーク内のエージェントプロセスの運動のそれぞれ直前及び直後においての本発明に従って形成されたエージェントプロセスの状態を示す図である。

【図36】オペレーション”go”の遂行による本発明のコンピュータネットワーク内のエージェントプロセスの運動のそれぞれ直前及び直後においての本発明に従って形成されたエージェントプロセスの状態を示す図である。

【図37】オペレーション”go”の遂行による本発明に従うネットワークを通るエージェントプロセスの運動のためのステップを示すフローチャートである。

【図38】パーミットを含むエージェントプロセスの構造を示す図である。

【図39】図38のパーミットの構造を示す図である。

【図40】オペレーション”go”の遂行による本発明に従うネットワークを通るエージェントプロセスの運動のためのステップを示すフローチャートである。

【図41】本発明に従って形成されたネットワークの構造を示すと共に、ネットワークを通るあるエージェントプロセスの運動を示す図である。

【図42】本発明の一実施例によるコード化エージェントの構造を示す図である。

【図43】オペレーション”go”の遂行による本発明に従うネットワークを通るエージェントプロセスの運動のためのステップを示すフローチャートである。

【図44】オペレーション”go”の遂行による本発明に

従うネットワークを通るエージェントプロセスの運動のためのステップを示すフローチャートである。

【図45】テレアドレス、サイテーション、テレネーム及びウェイを含む図35のチケットの構造を示す図である。

【図46】図45のウェイの構造を示す図である。

【図47】本発明の一実施例によって形成されたディクショナリを示す図である。

【図48】本発明の一実施例によって形成されたファイндаの構造を示す図である。

【図49】オペレーション”go”の遂行による本発明に従うネットワークを通るエージェントプロセスの運動のためのステップを示すフローチャートである。

【図50】本発明に従って形成されたネットワークの構造を示すと共に、ネットワークを通るあるエージェントプロセスの運動を示す図である。

【図51】本発明に従って形成されたネットワークの構造を示すと共に、ネットワークを通るあるエージェントプロセスの運動を示す図である。

【図52】オペレーション”go”の遂行による本発明に従うネットワークを通るエージェントプロセスの運動のためのステップを示すフローチャートである。

【図53】本発明に従って形成されたネットワークの構造を示すと共に、ネットワークを通るあるエージェントプロセスの運動を示す図である。

【図54】オペレーション”go”の遂行による本発明に従うネットワークを通るエージェントプロセスの運動のためのステップを示すフローチャートである。

【図55】本発明の一実施例によるオペレーション”entering”の動作の遂行においてなされるステップを示すフローチャートである。

【図56】それぞれオペレーション”entering”の直前及び直後のエージェントプロセスの状態を示す図である。

【図57】それぞれオペレーション”entering”の直前及び直後のエージェントプロセスの状態を示す図である。

【図58】オペレーション”exiting”の遂行のそれぞれ直前及び直後のプロセスの状態の一部を示す図である。

【図59】オペレーション”exiting”の遂行のそれぞれ直前及び直後のプロセスの状態の一部を示す図である。

【図60】本発明の一局面に従うネットワークを通るエージェントプロセスの移動についての各ステップを示すフローチャートである。

【図61】本発明の一局面に従うネットワークを通るエージェントプロセスの移動についての各ステップを示すフローチャートである。

【図62】図60～図62に示したステップにおいて使

用されるインターチェンジ（相互変換）されたオブジェクトのレポジトリの構造を示す。

【図63】本発明の一局面に従うネットワークを通るエージェントプロセスの移動についての各ステップを示すフローチャートである。

【図64】本発明の原理によって形成されたコンピュータネットワークを示す図である。

【図65】それぞれオペレーション”send”の遂行の直前及び直後のエージェントプロセスの状態の一部を示す図である。

【図66】それぞれオペレーション”send”の遂行の直前及び直後のエージェントプロセスの状態の一部を示す図である。

【図67】オペレーション”send”の直後においてのエージェントプロセスのクロンの状態を示す図である。

【図68】オペレーション”send”の遂行においてエージェントプロセスのクロンの形成のそれぞれ前後においてのエンジンプロセスの状態を示す図である。

【図69】オペレーション”send”の遂行においてエージェントプロセスのクロンの形成のそれぞれ前後においてのエンジンプロセスの状態を示す図である。

【図70】オペレーション”send”の遂行においてエージェントプロセスの各クロンがコンピュータネットワークのそれぞれのコンピュータシステムに移行した、図25のコンピュータネットワークを示す図である。

【図71】デифァードクローニングと称される本発明の別の局面に従ってあるエージェントプロセスのクロンがそれを通して移動するコンピュータネットワークを示す図である。

【図72】ニル（零）及びチケットリストを含む送フレームの構造を示す図である。

【図73】本発明の一局面に従って形成されたエージェントプロセスのコード化クロンの構造を示す図である。

【図74】デифァードクローニングと称される本発明の別の局面に従ってあるエージェントプロセスのクロンがそれを通して移動するコンピュータネットワークを示す図である。

【図75】デифァードクローニングと称される本発明の別の局面に従ってあるエージェントプロセスのクロンがそれを通して移動するコンピュータネットワークを示す図である。

【図76】デифァードクローニングと称される本発明の別の局面に従ってあるエージェントプロセスのクロンがそれを通して移動するコンピュータネットワークを示す図である。

【図77】デифァードクローニングと称される本発明の別の局面に従ってあるエージェントプロセスのクロンがそれを通して移動するコンピュータネットワークを示す図である。

【図78】オペレーション”meet”の遂行のそれぞれ直前及び直後においてのあるプロセスの状態を示す図である。

【図79】オペレーション”meet”の遂行のそれぞれ直前及び直後においてのあるプロセスの状態を示す図である。

【図80】オペレーション”meet”の遂行においてとられるステップのフローチャートである。

【図81】コンタクトのセットを含むエージェントプロセスの状態の一部を示す図である。

【図82】オペレーション”meeting”の遂行のそれぞれ直前及び直後においてのあるプロセスの状態を示す図である。

【図83】本発明の一実施例に従うオペレーション”meeting”の遂行においてとられる各ステップを示すフローチャートである。

【図84】オペレーション”meeting”の遂行のそれぞれ直前及び直後においてのあるプロセスの状態を示す図である。

【図85】本発明の一局面に従って相互作用する2つのエージェントプロセスの状態を示す図である。

【図86】オペレーション”part”の遂行のそれぞれ直前及び直後においてのあるエージェントプロセスの状態の一部を示す図である。

【図87】オペレーション”part”の遂行のそれぞれ直前及び直後においてのあるエージェントプロセスの状態の一部を示す図である。

【図88】オペレーション”part”の遂行の直後において、図85に示したエージェントプロセスの状態を示す図である。

【図89】本発明による第2のエージェントプロセスとの相互作用の間の第1のエージェントプロセスの状態の一部を示す図である。

【図90】本発明による第2のエージェントプロセスとの相互作用の間の第1のエージェントプロセスの状態の一部を示す図である。

【図91】本発明による第2のエージェントプロセスとの相互作用の間の第1のエージェントプロセスの状態の一部を示す図である。

【図92】本発明による第2のエージェントプロセスとの相互作用の間の第1のエージェントプロセスの状態の一部を示す図である。

【図93】本発明による第2のエージェントプロセスとの相互作用の間の第1のエージェントプロセスの状態の一部を示す図である。

【図94】本発明による第2のエージェントプロセスとの相互作用の間の第1のエージェントプロセスの状態の一部を示す図である。

【図95】本発明の一実施例による図89～94に示す相互作用の間の第2のエージェントプロセスの各行程を

示すフローチャートである。

【図96】本発明の一局面に従うクラス定義の構造を示す図である。

【図97】本発明の一実施例に従って形成されたクラスオブジェクトの構造を示す図である。

【図98】本発明の第2の実施例に従って形成されたクラスオブジェクトの構造を示す図である。

【図99】本発明に従って形成されたインターフェースオブジェクトの構造を示す図である。

【図100】セット、識別子及びブーリアンを含むフィーチャ定義の構造を示す図である。

【図101】コンストレイント及びブーリアンを含む属性定義の構造を示す図である。

【図102】コンストレイントを及びリスト（コンストレイントを含む）を含むオペレーション定義の構造を示す図である。

【図103】コンストレイントの構造を示す図である。

【図104】2つのリスト及び6つのレキシコンを含むインプリメンテーションオブジェクトの構造を示す図である。

【図105】プロシージャ及びリスト（識別子を含む）を含む方法の構造を示す図である。

【図106】図105のプロシージャの構造を示す図である。

【図107】サイティイション（テレネーム、2つのインテジャ（整数）及び識別子を含む）の構造を示す図である。

【図108】サイティイションをそれぞれ含む2つの引用（サイト）されたオブジェクトの構造を示す図である。

【図109】オペレーション”do”の遂行においての各ステップを示すフローチャートである。

【図110】1つの整数及び1つのプロシージャを含む予め定義されたフレームの構造を示す図である。

【図111】スタック（ユーザーフレームを含む）を含むプロセスの実行状態を示す図である。

【図112】図111のユーザーフレームの状態を示す図である。

【図113】ユーザ定義されたオペレーションの遂行においての各ステップを示す単一のフローチャートである。

【図114】ユーザ定義されたオペレーションの遂行においての各ステップを示す単一のフローチャートである。

【図115】スタック（フレームを含む）を含むプロセスの実行状態を示す図である。

【図116】第2のフレーム及び図115のプロセスの実行状態を示す図である。

【図117】最初に述べたフレームと第2のフレームとを含むスタックを含む図115、図116のプロセスの

実行状態を示す図である。

【図118】本発明に従うクラス階層からの方法及びフィーチャ定義を選択する上の各ステップを示すフローチャートである。

【図119】リストがそのメンバーであるクラスの階層を示す図である。

【図120】リストクラスがメンバーであるクラス（複数）の階層を示す図である。

【図121】あるオブジェクトのイニシャライゼーションのための各ステップを示すフローチャートである。

【図122】オペレーション”if”の遂行のための各ステップを示すフローチャートである。

【図123】オペレーション”either”の遂行のための各ステップを示すフローチャートである。

【図124】オペレーション”select”の遂行のための各ステップを示すフローチャートである。

【図125】オペレーション”select”の遂行の直後におけるプロセスの実行状態を示す図である。

【図126】オペレーション”select”の遂行の直後におけるプロセスの実行状態を示す図である。

【図127】オペレーション”select”の遂行の動的状態を示すあらかじめ定義されたフレームの状態を示す図である。

【図128】オペレーション”while”の遂行においてとられるステップを示すフローチャートである。

【図129】オペレーション”catch”のそれぞれ直前及び直後においてのあるプロセスの実行状態の一部を示す図である。

【図130】オペレーション”catch”のそれぞれ直前及び直後においてのあるプロセスの実行状態の一部を示す図である。

【図131】オペレーション”catch”の遂行のための各ステップを示すフローチャートである。

【図132】オペレーション”loop”の遂行のための各ステップを示すフローチャートである。

【図133】実行されたオブジェクトと2つの整数とを含むレピートフレームの構造を示す図である。

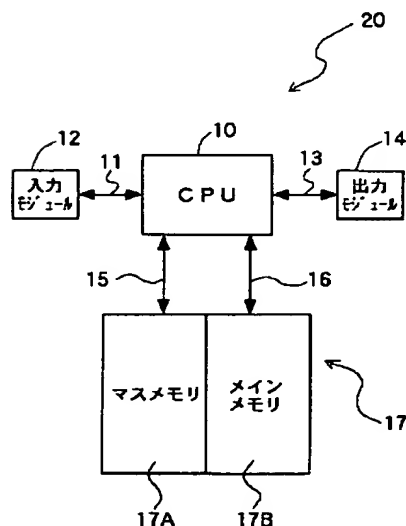
【図134】オペレーション”repeat”の遂行のための各ステップを示すフローチャートである。

【図135】スタック（ユーザー定義フレーム、予定義フレーム、レピートフレーム及び第2のユーザー定義フレームを上部から下部にかけて含む。）を含むプロセスの実行状態を示す図である。

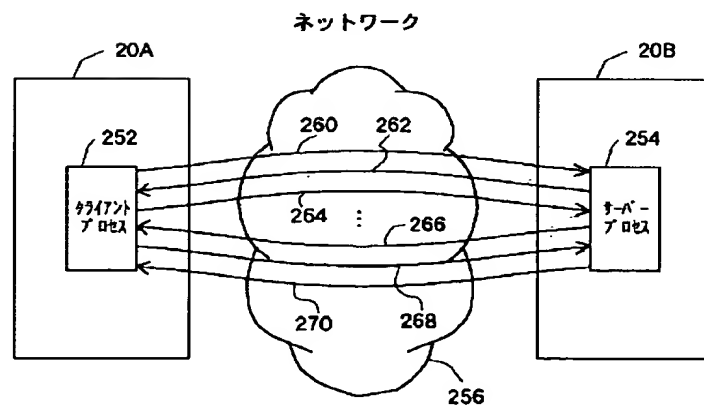
【符号の説明】

100 コンピュータネットワーク  
102AB コミュニケーションリンク  
120A、120B コンピュータシステム  
150A エージェント  
220A、220B プレイス  
1306 チケット

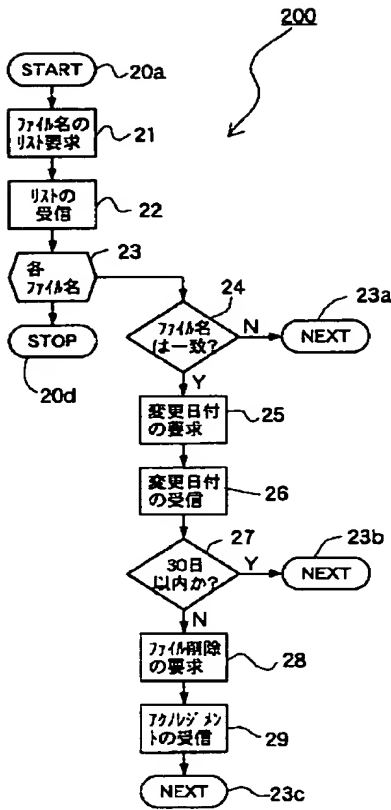
【図1】



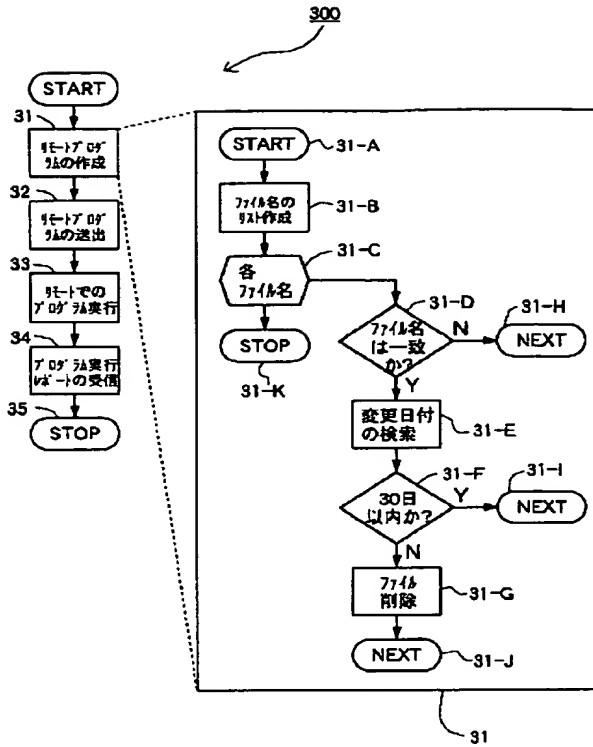
【図3】



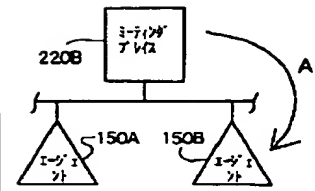
【図2】



【図4】

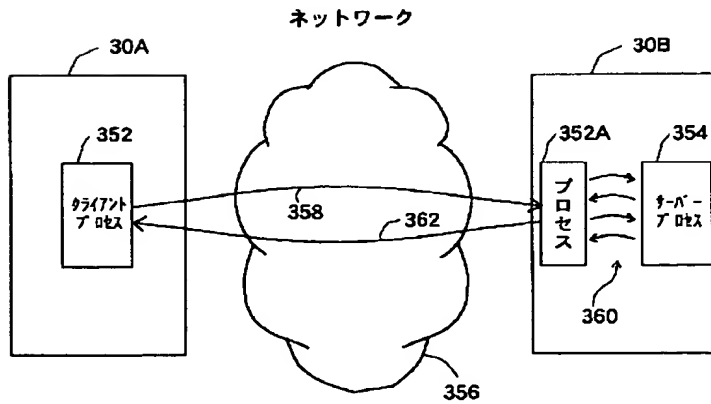


【図21】

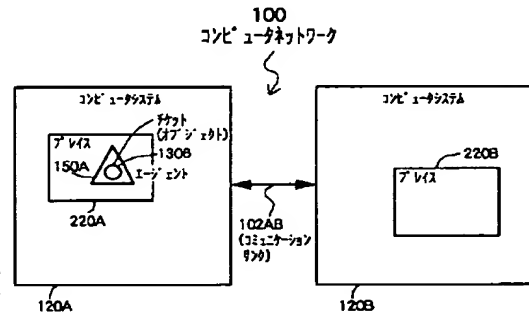


リモートプログラムのフローチャート

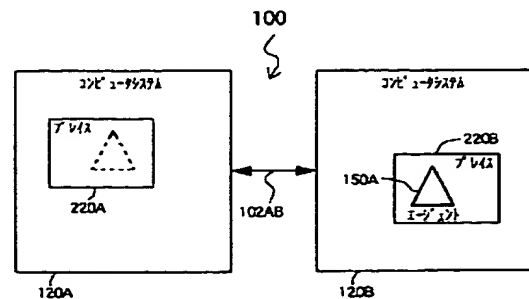
【図5】



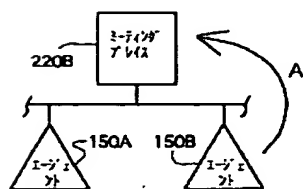
【図6】



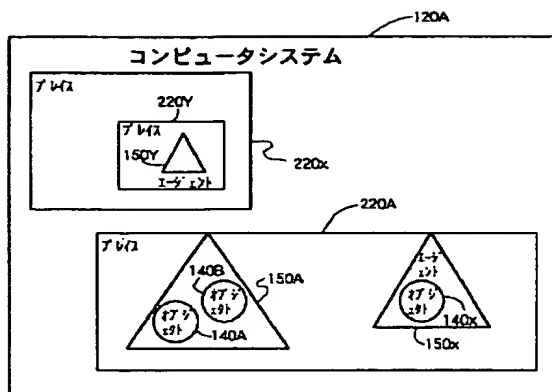
【図7】



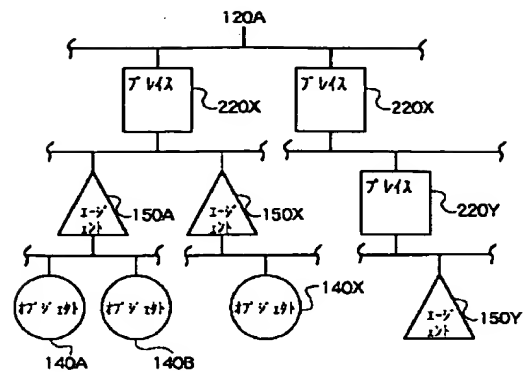
【図22】



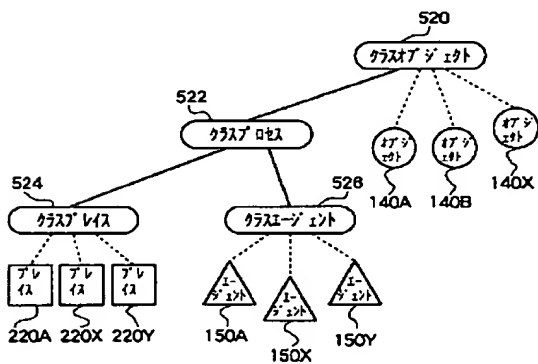
【图8】



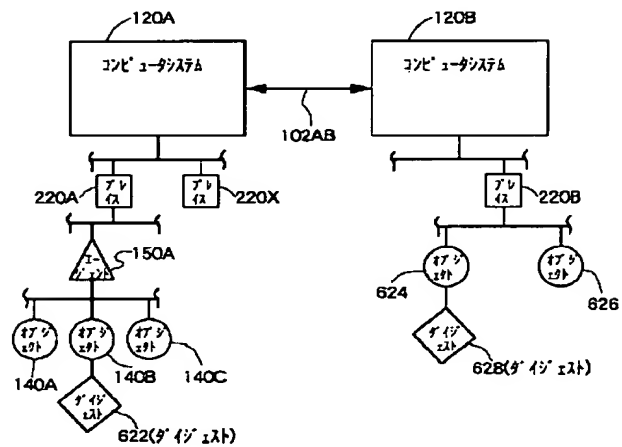
【図 9】



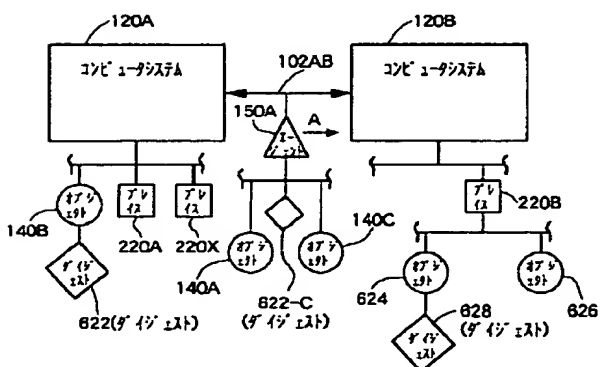
【図 10】



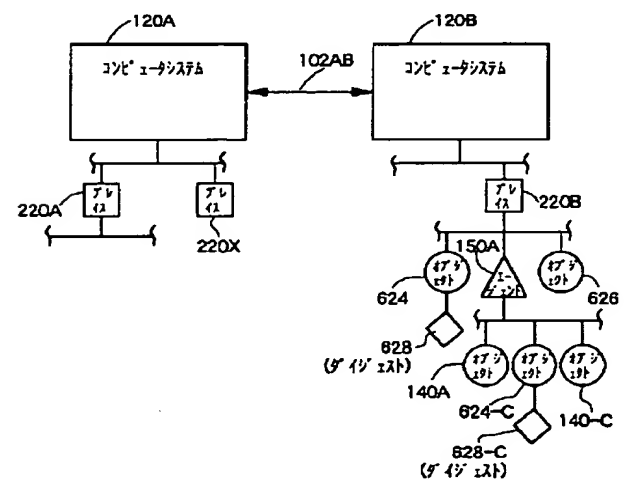
【図 1 1】



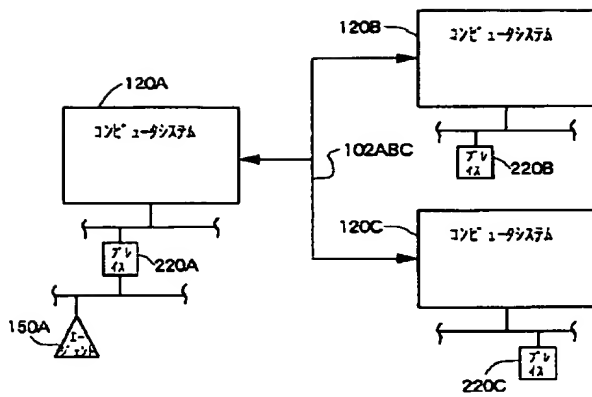
【图 12】



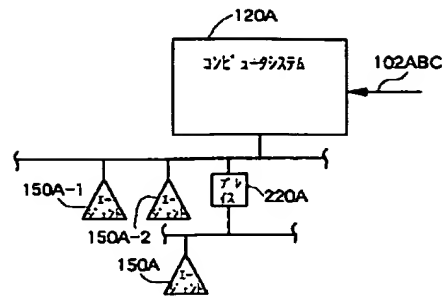
【图 13】



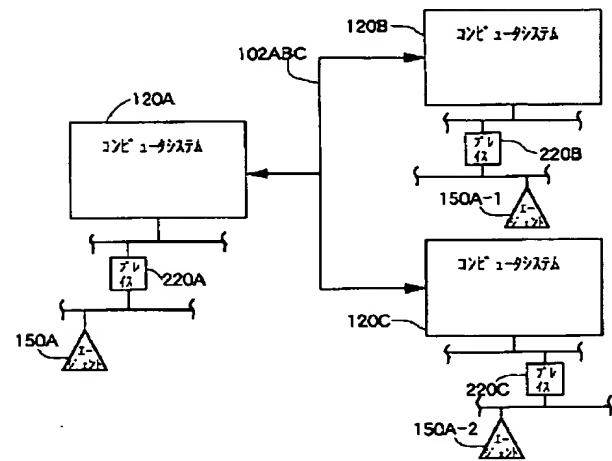
【図14】



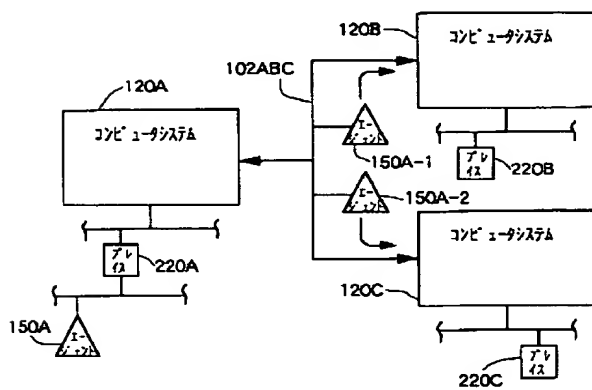
【図15】



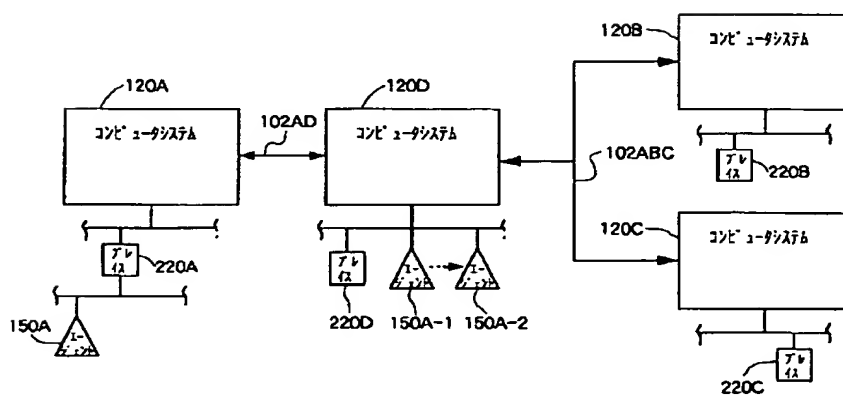
【図17】



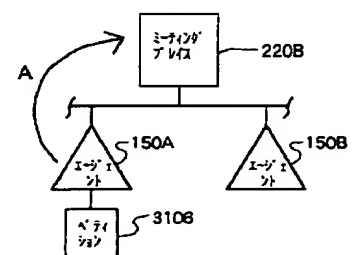
【図16】



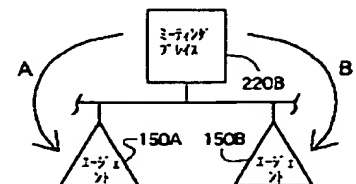
【図18】



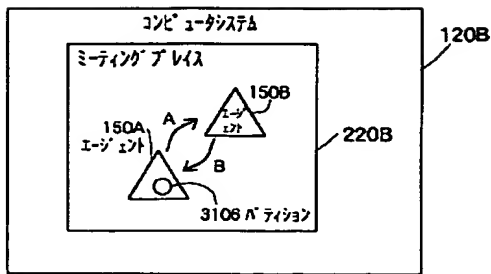
【図20】



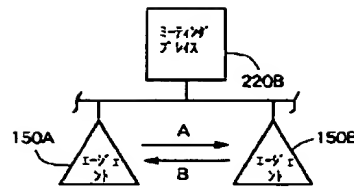
【図23】



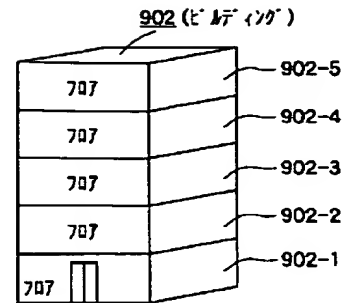
【図19】



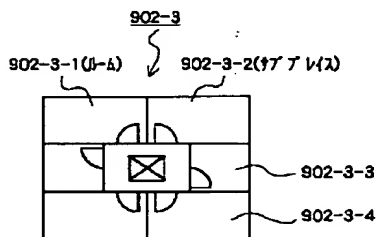
【図24】



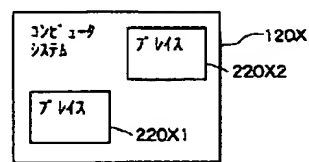
【図25】



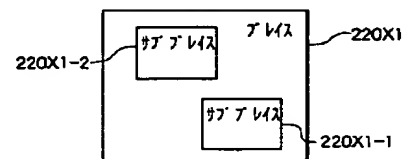
【図26】



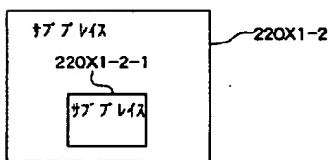
【図27】



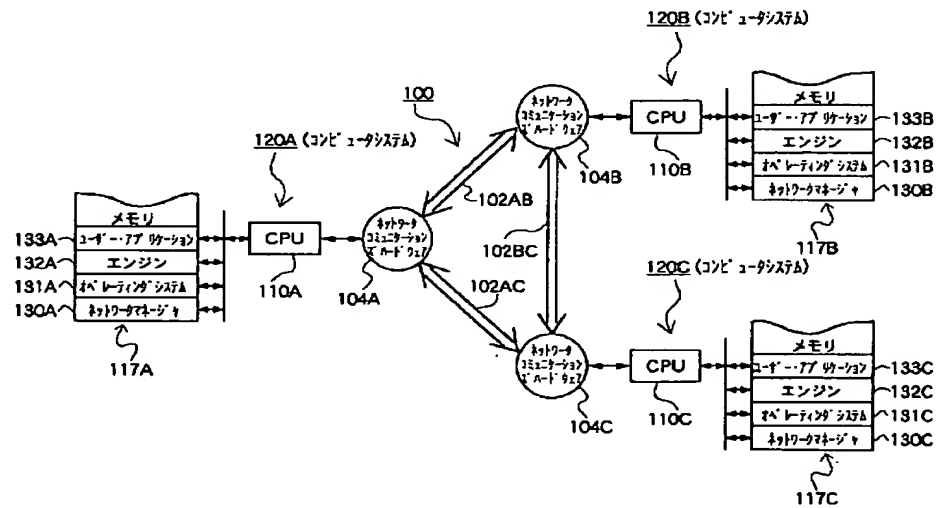
【図28】



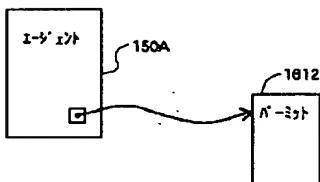
【図29】



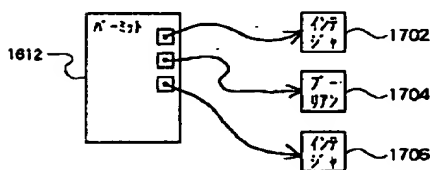
【図30】



【図38】

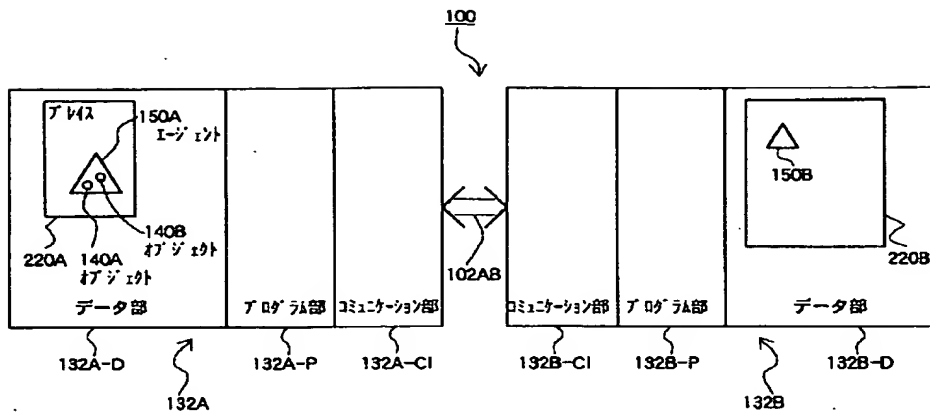


【図39】

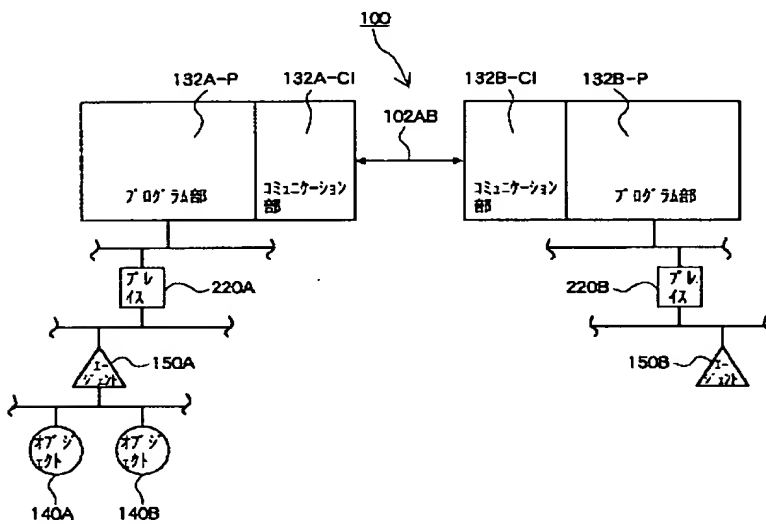




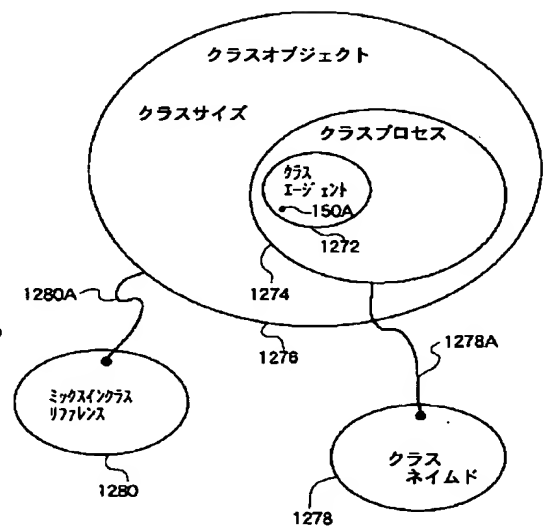
【図 3 1】



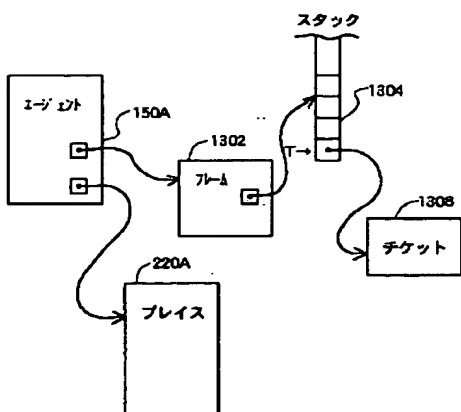
【図 3 2】



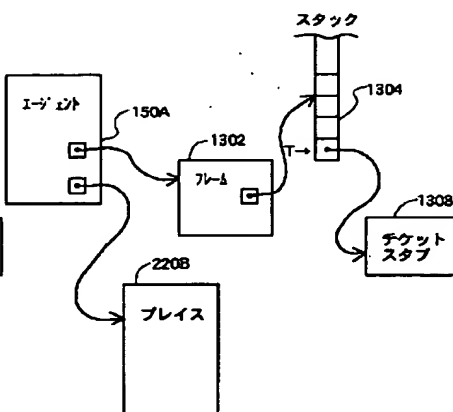
【図 3 3】



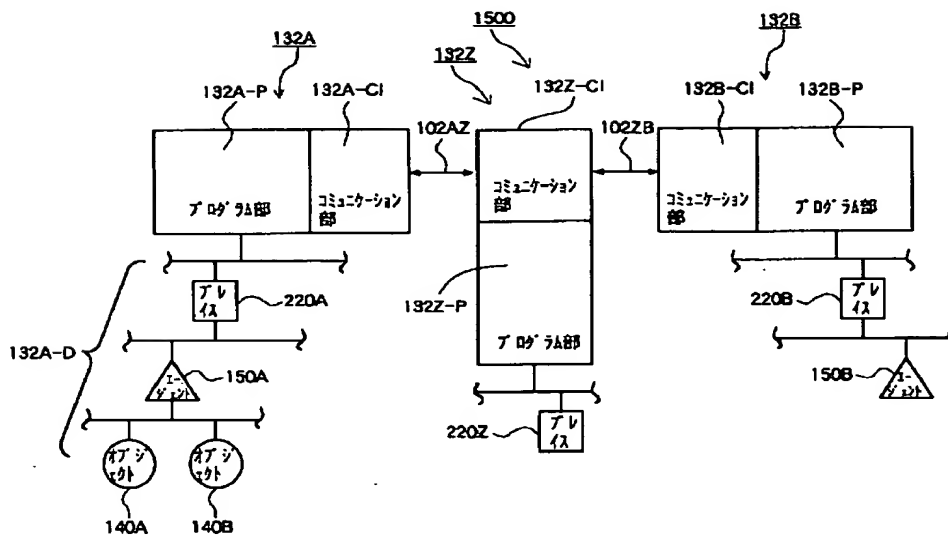
【図 3 5】



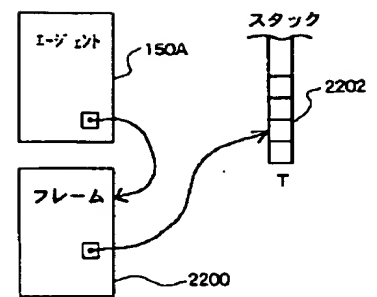
【図 3 6】



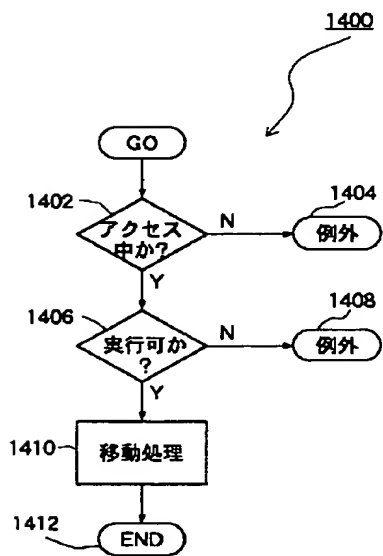
【図34】



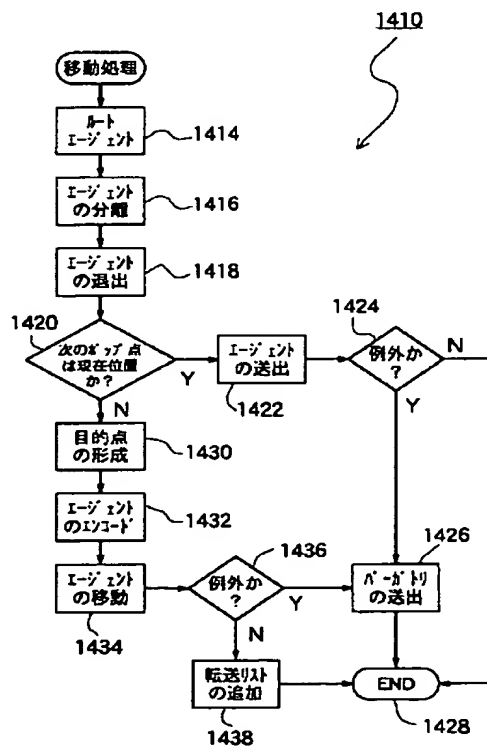
【図56】



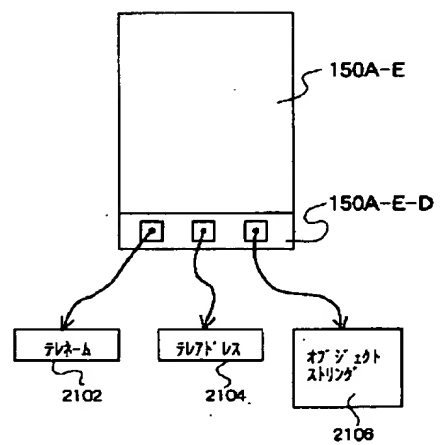
【図37】



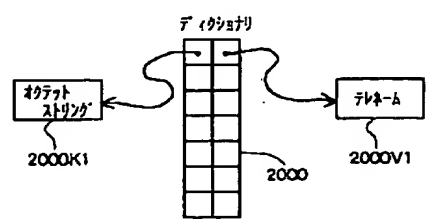
【図40】



【図42】



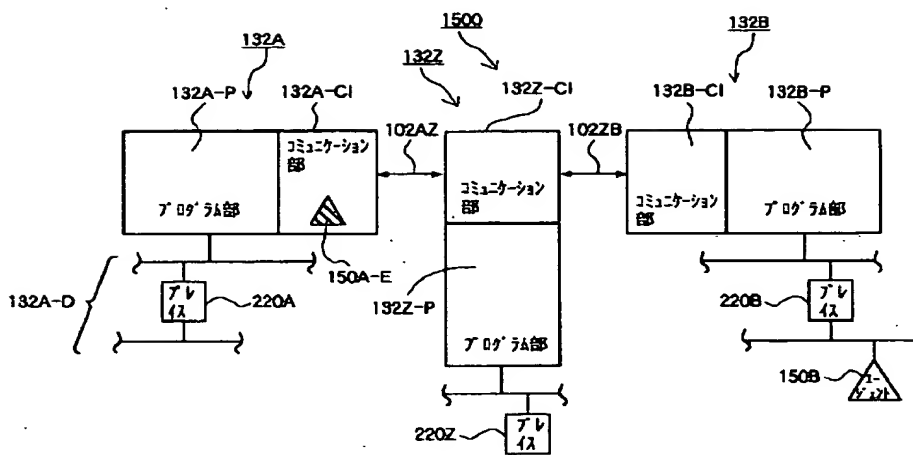
【図47】



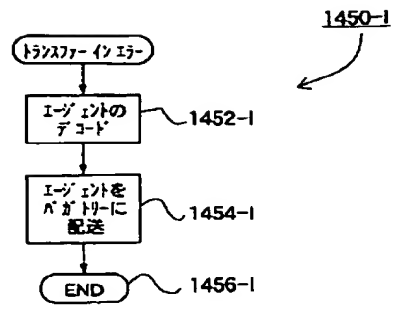
【図46】



【図41】

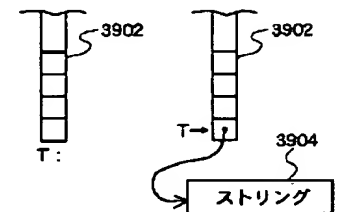


【図54】

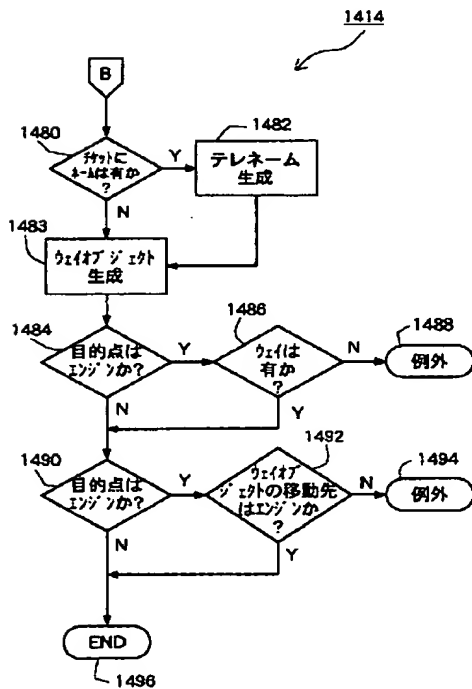


【図89】

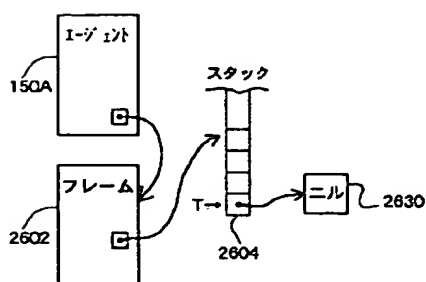
【図90】



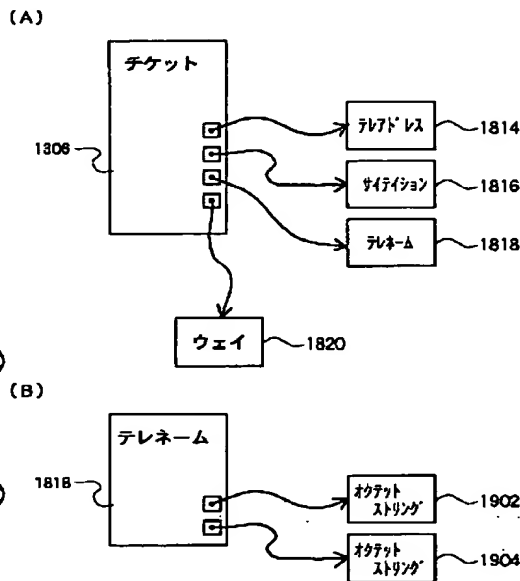
【図44】



【図66】

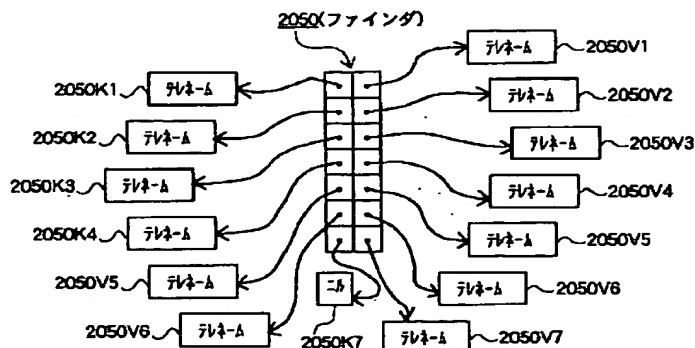
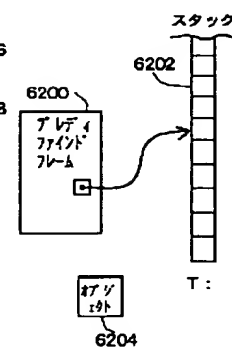


【図45】

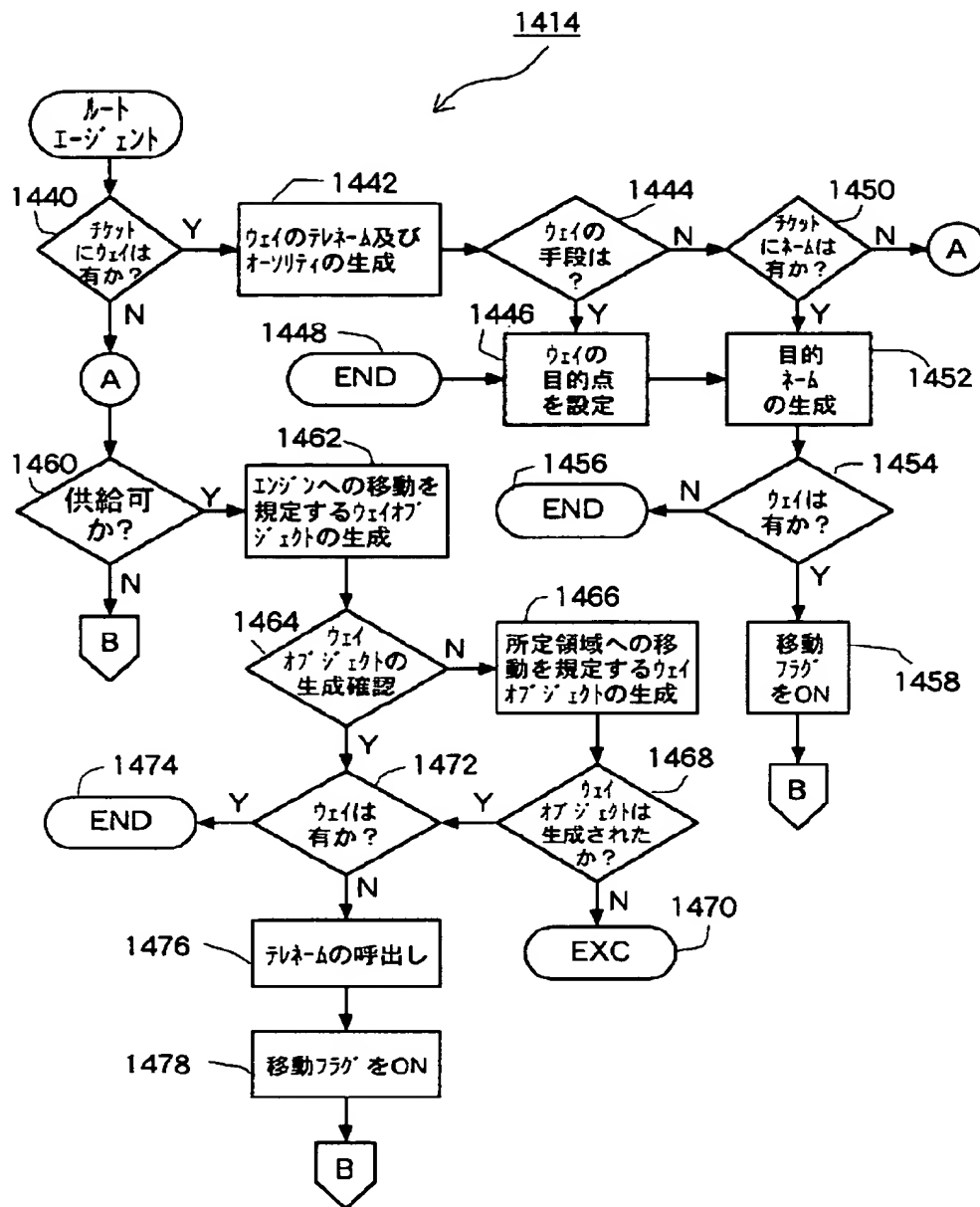


【図48】

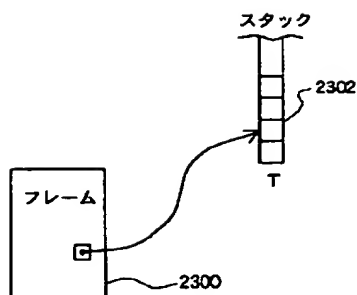
【図126】



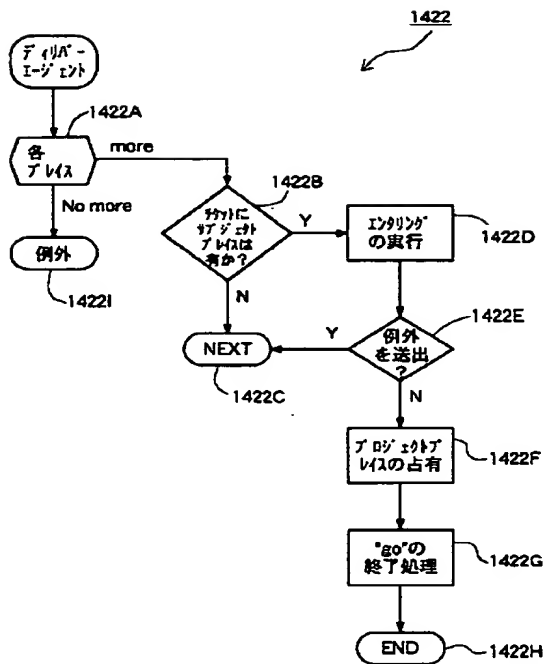
【図43】



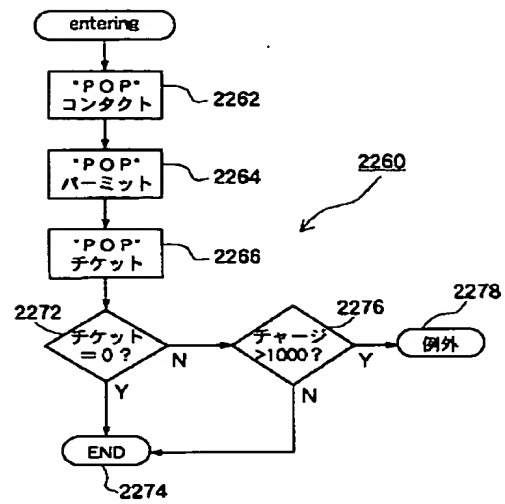
【図58】



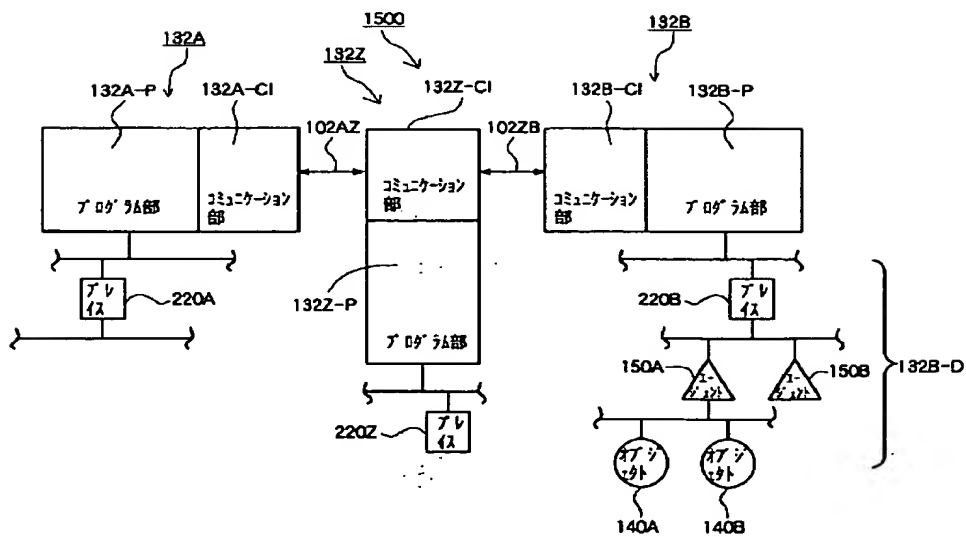
【図49】



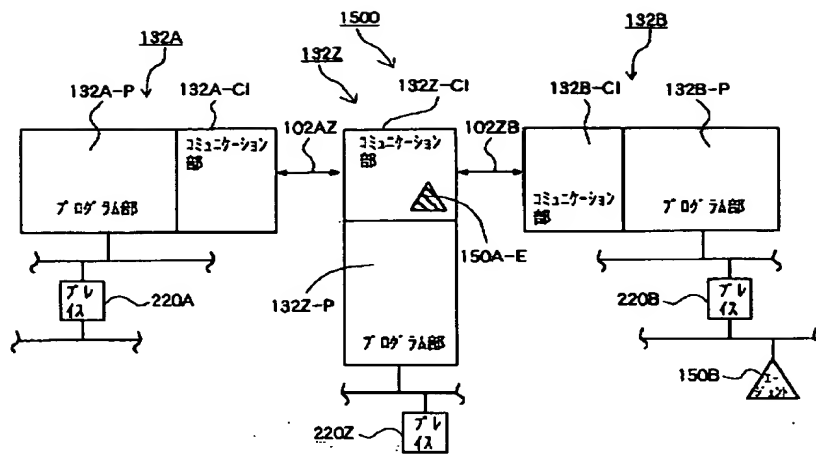
【図55】



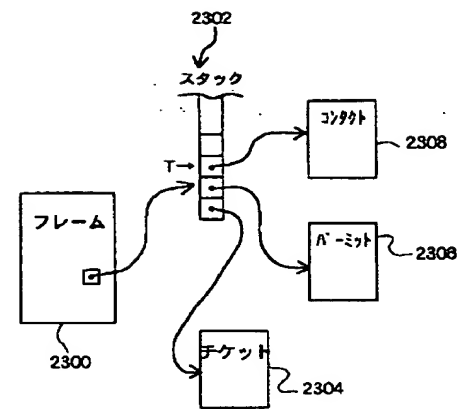
【図50】



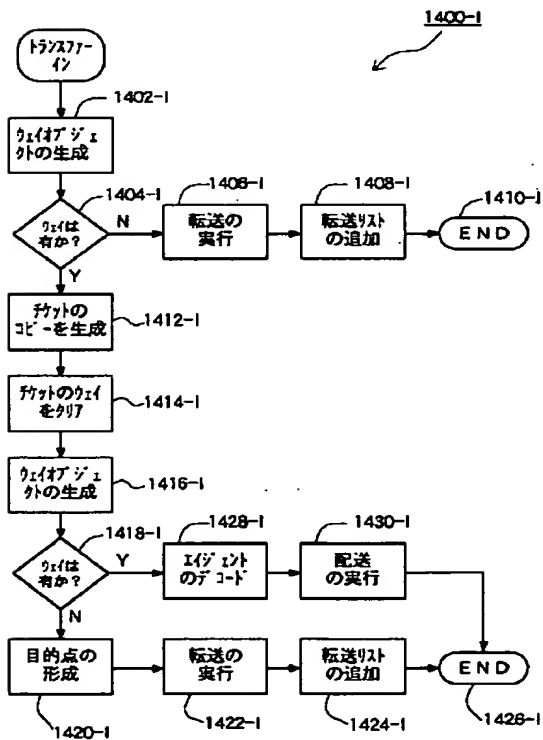
【図51】



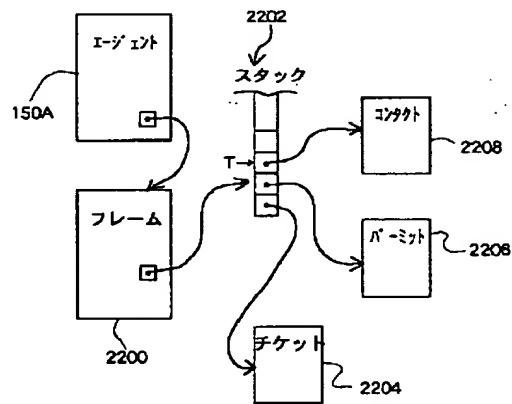
【図57】



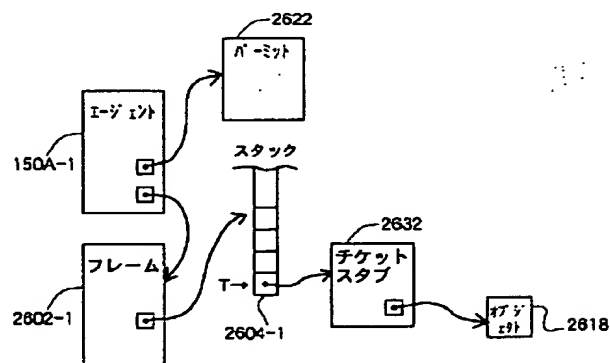
【図52】



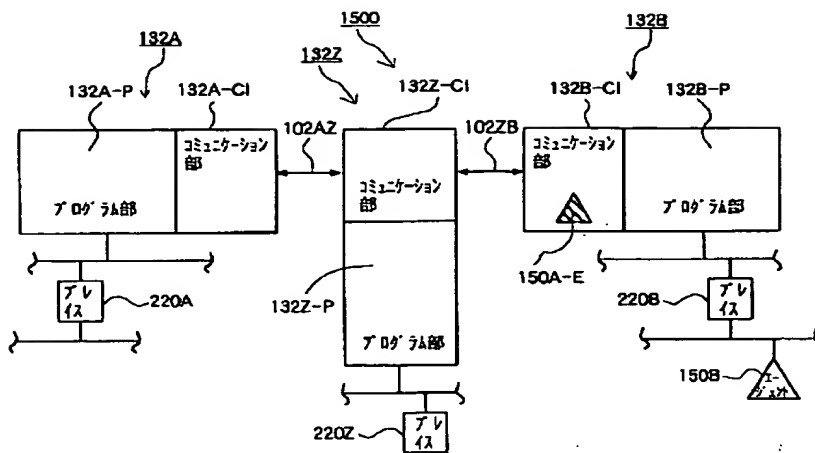
【図59】



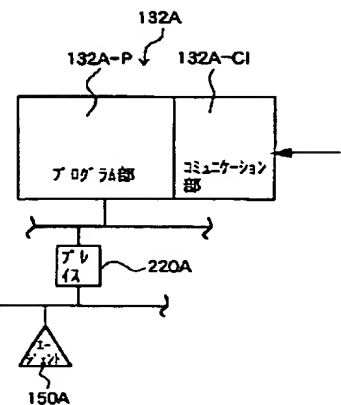
【図67】



【図53】

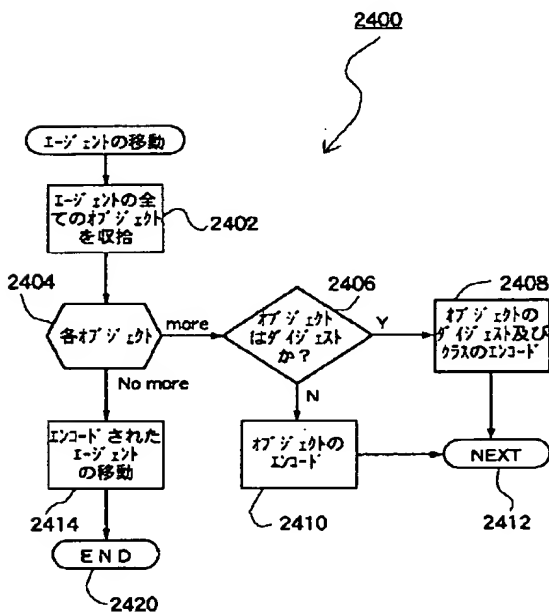


【図68】

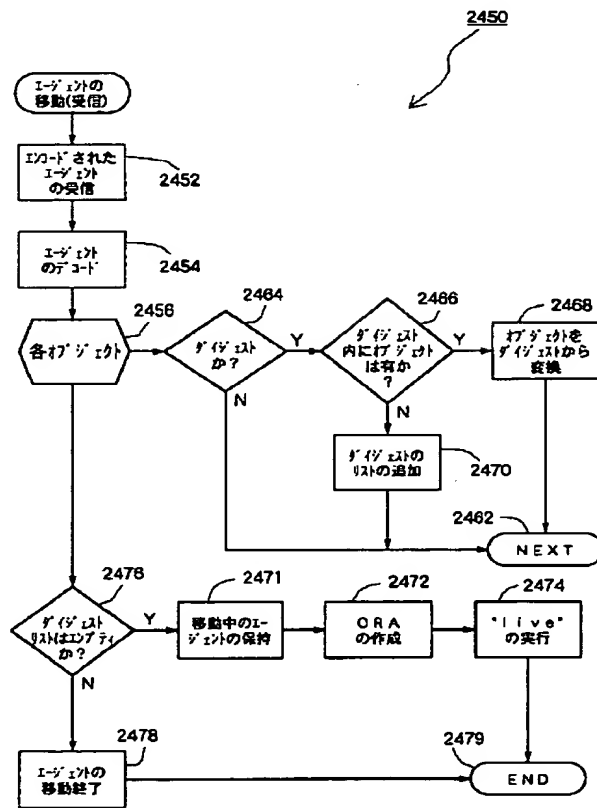
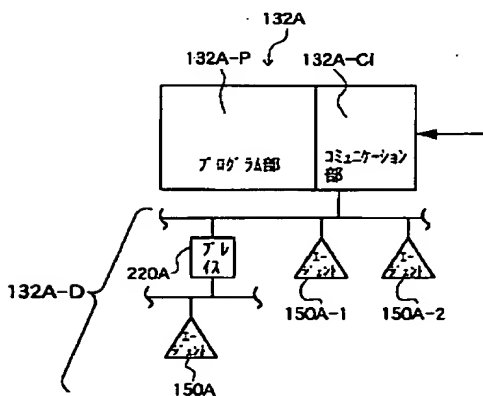


【図60】

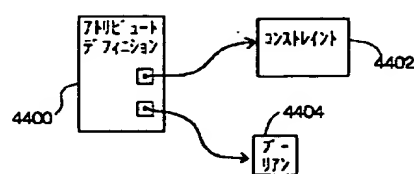
【図61】



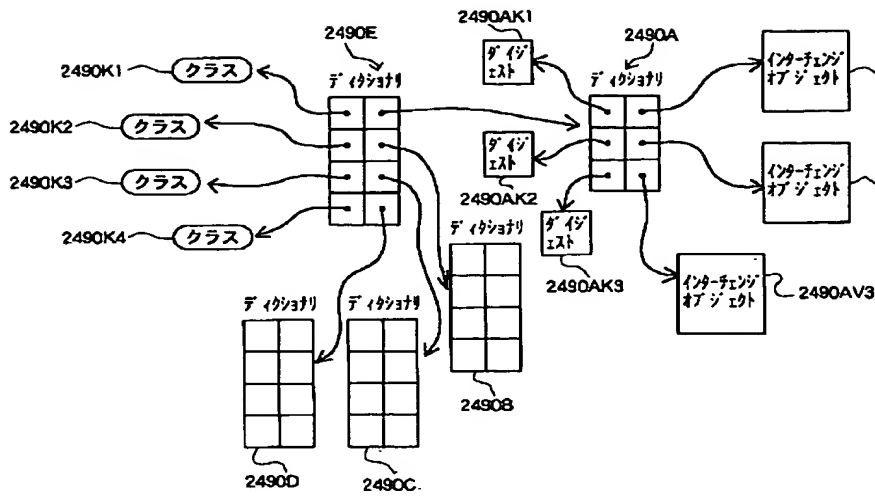
【図69】



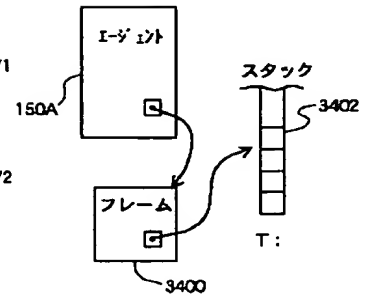
【図101】



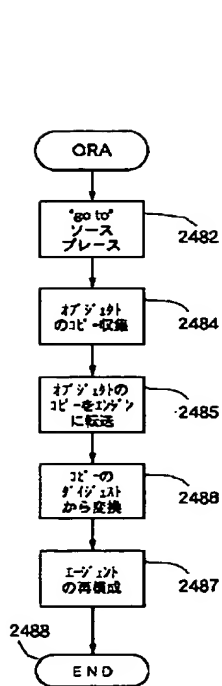
【図62】



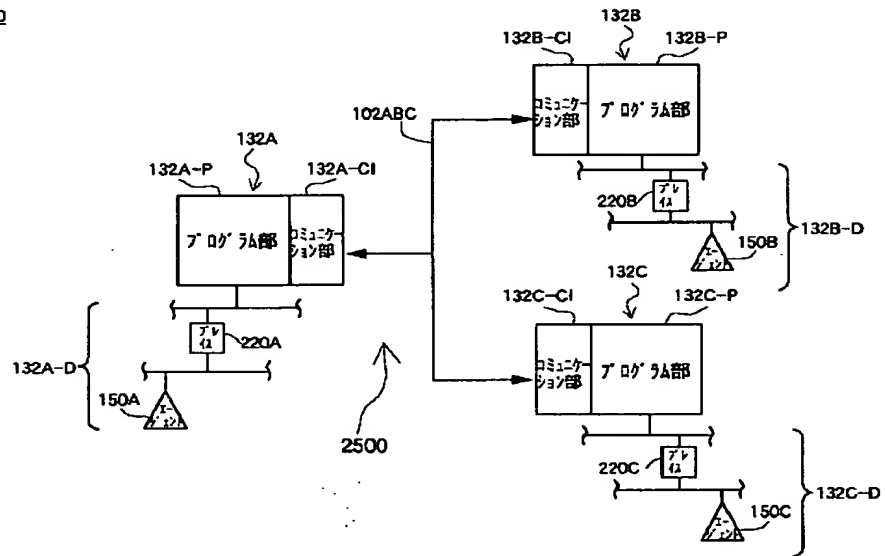
【図84】



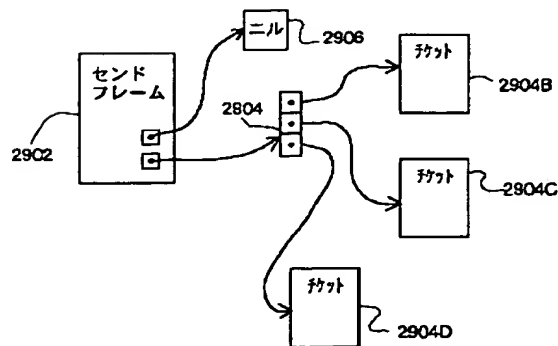
【図63】



【図64】

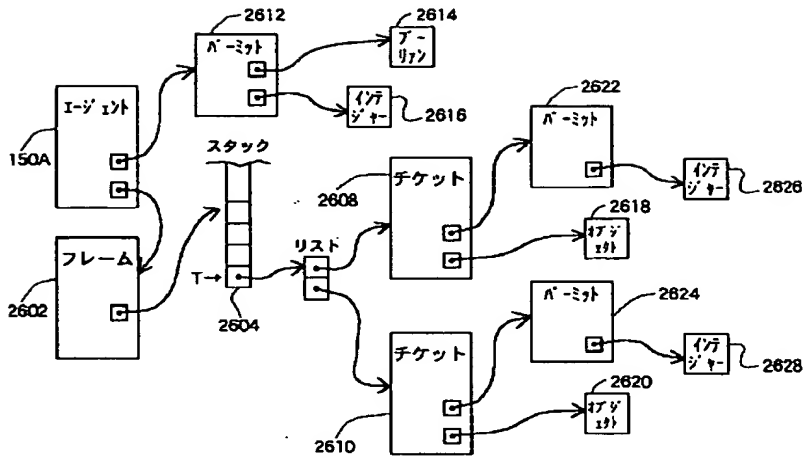


【図72】

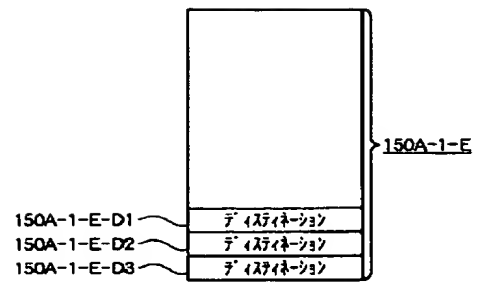




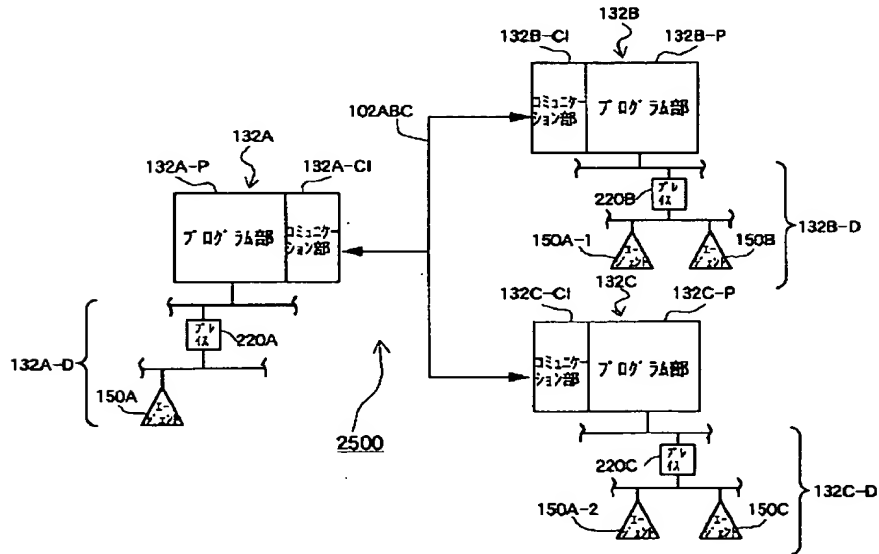
【図65】



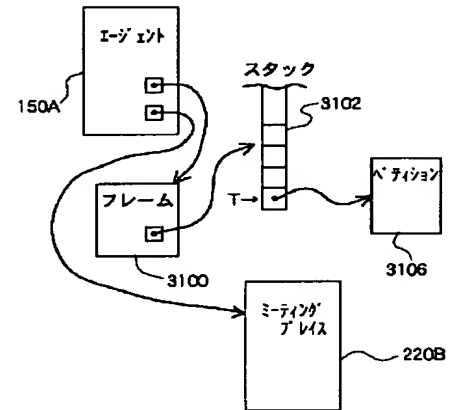
【図73】



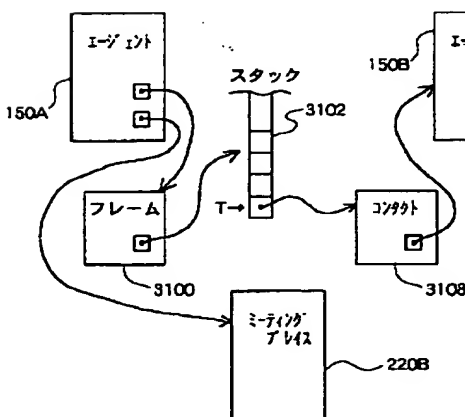
【図70】



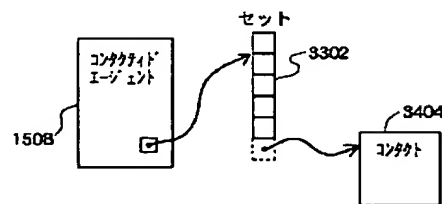
【図78】



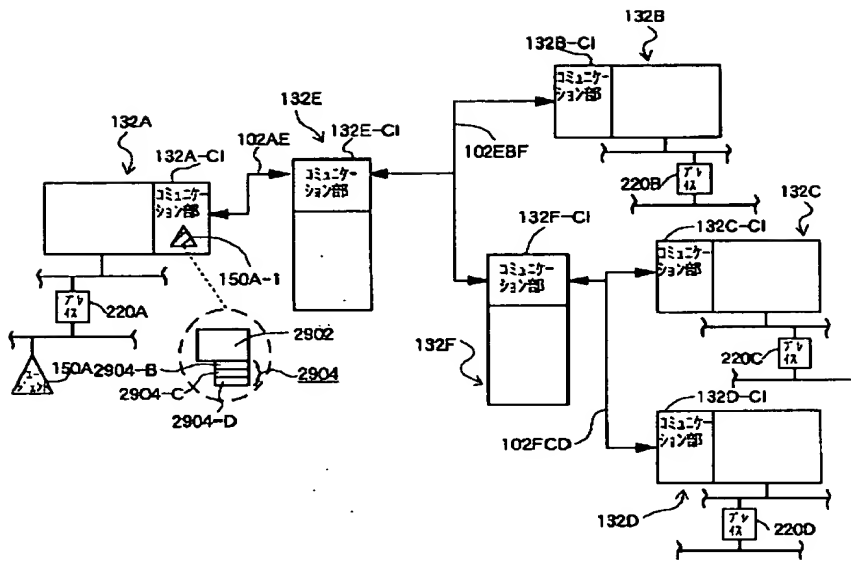
【図79】



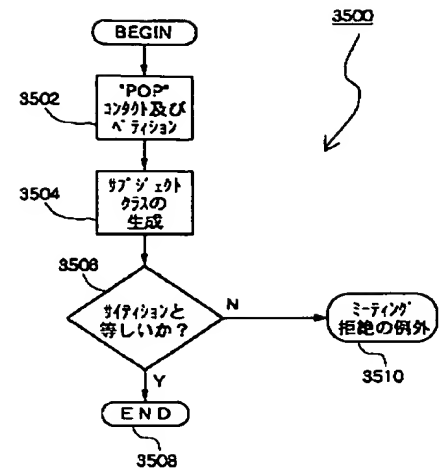
【図81】



【図 7 1】

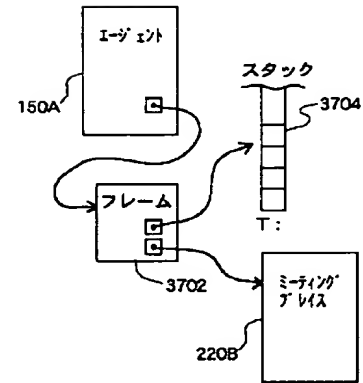
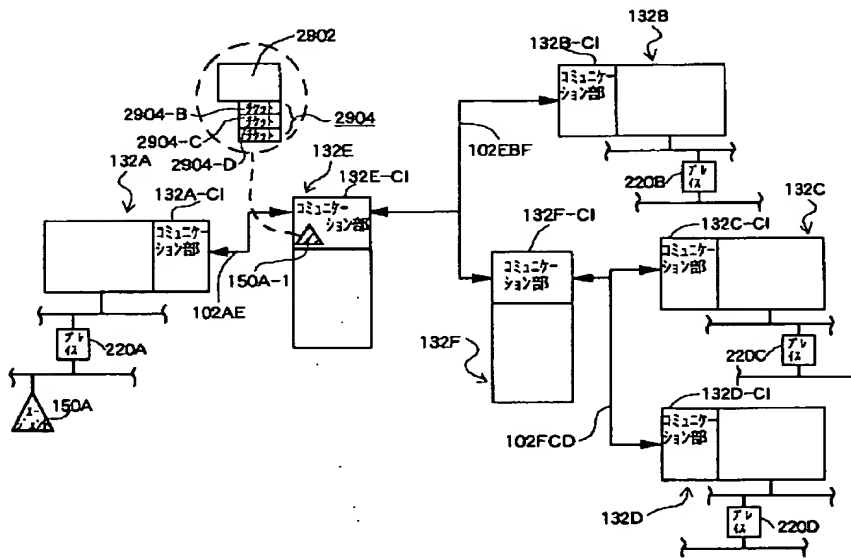


【図 8 3】



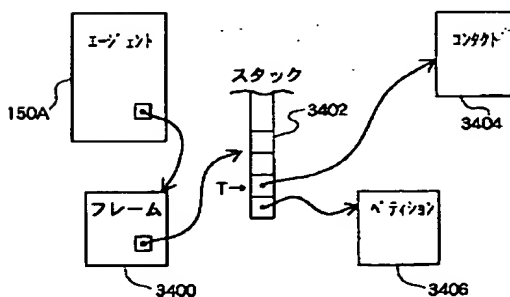
【図 8 7】

【図 7 4】

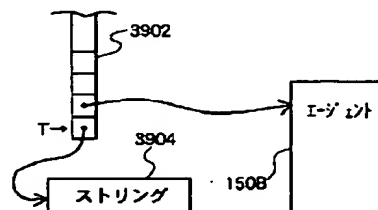


【図 9 2】

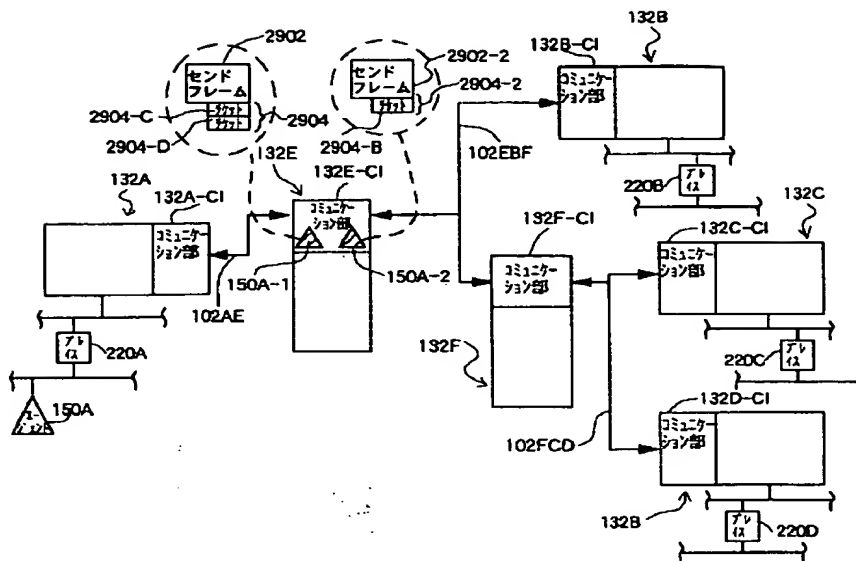
【図 8 2】



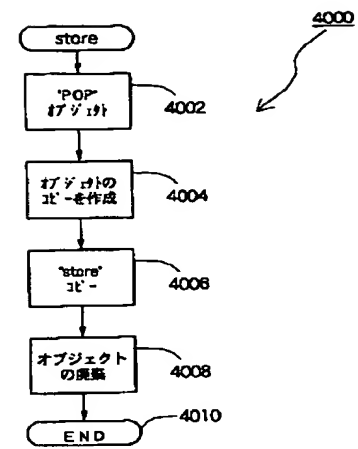
【図 9 1】



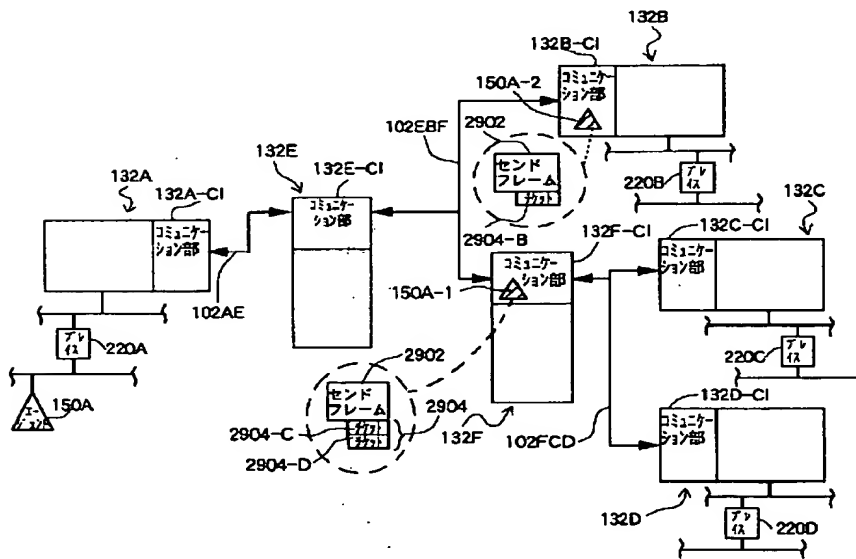
【図 7 5】



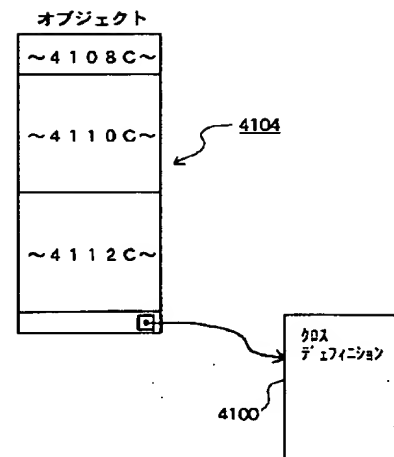
【図 9 5】



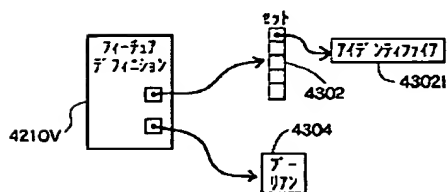
【図 7 6】



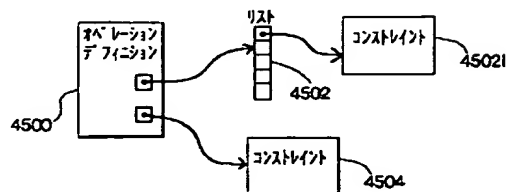
【図 9 8】



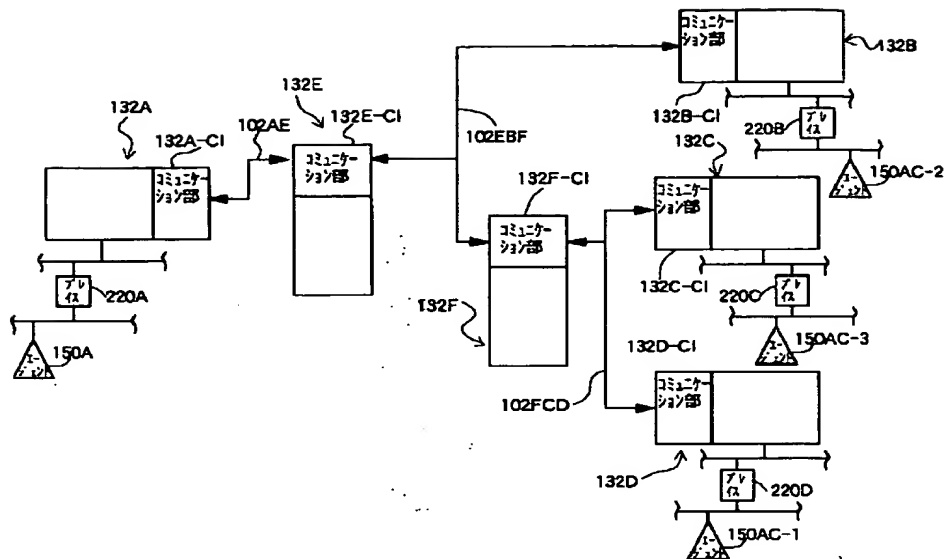
【図 10 0】



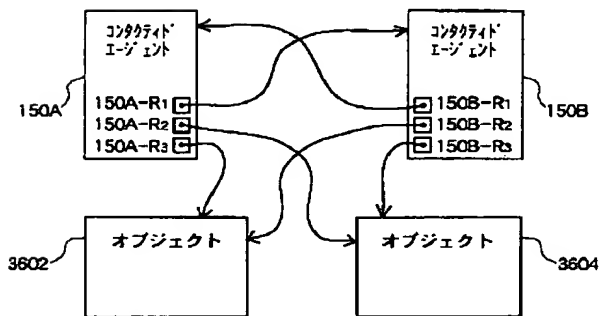
【図 10 2】



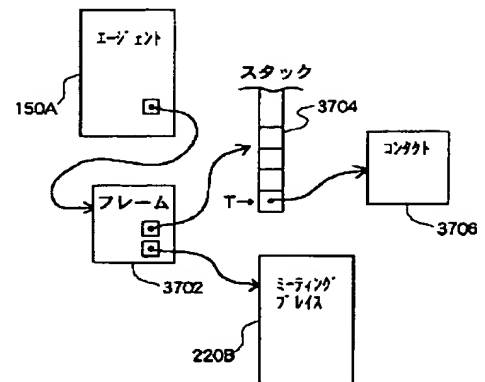
【図 7 7】



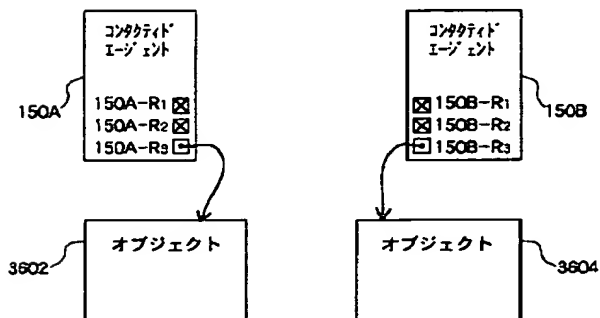
【図 8 5】



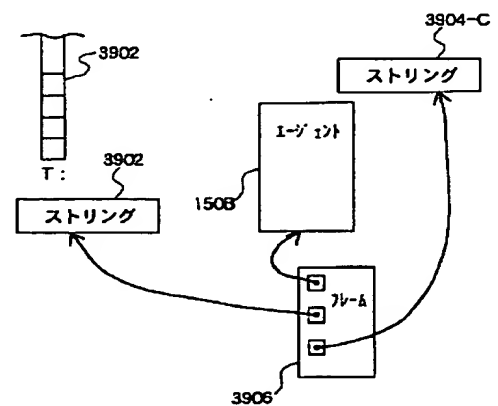
【図 8 6】



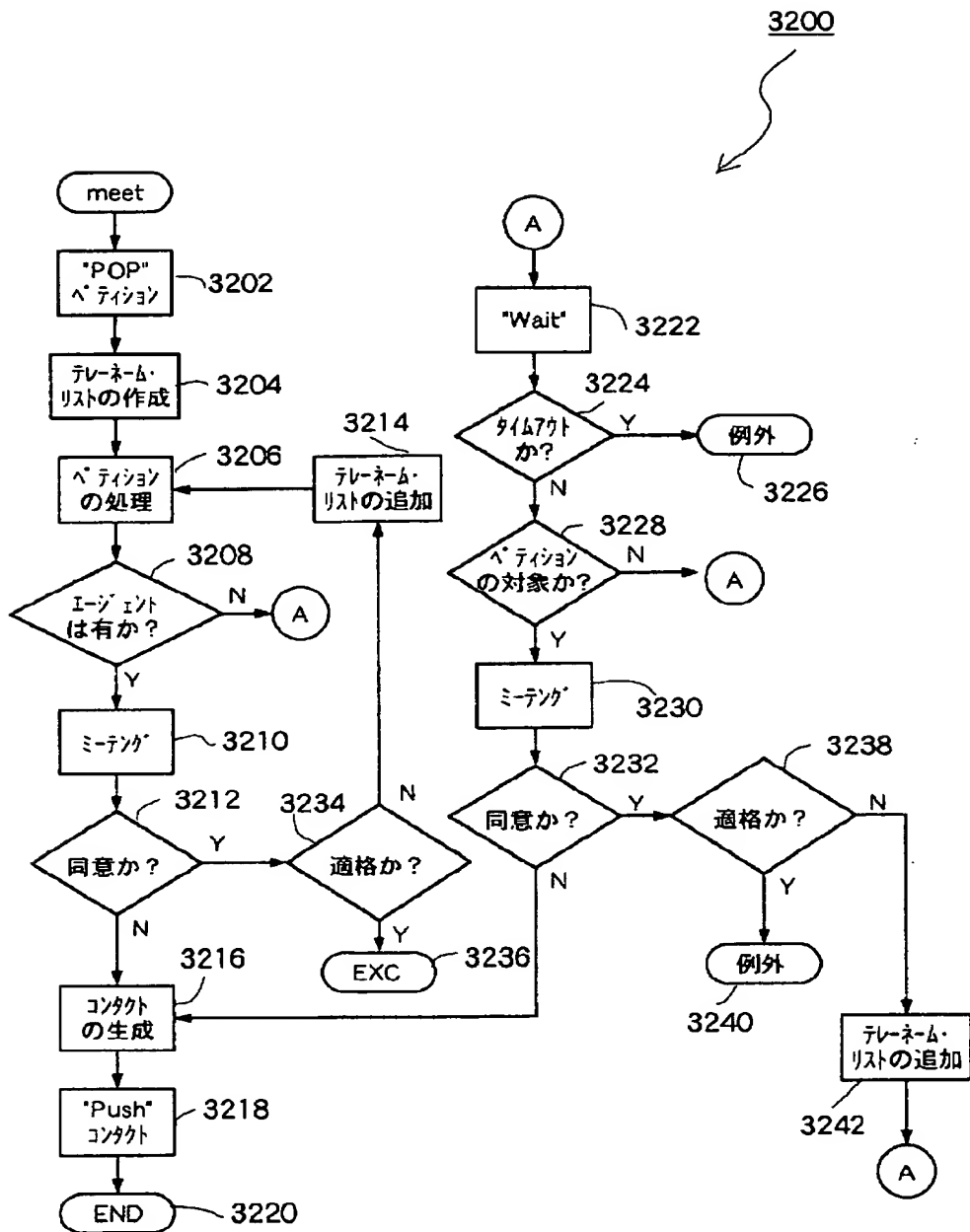
【図 8 8】



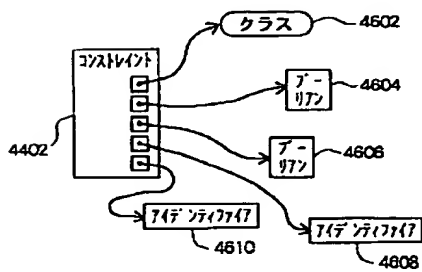
【図 9 3】



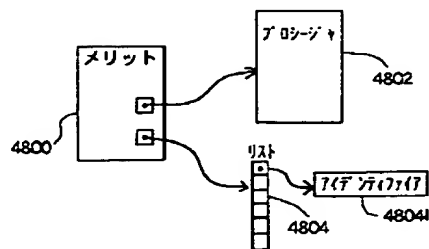
【図80】



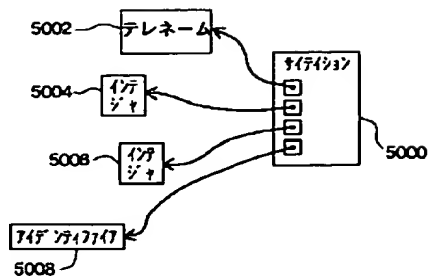
【図103】



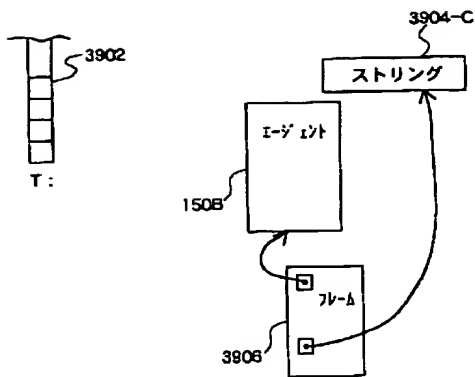
【図105】



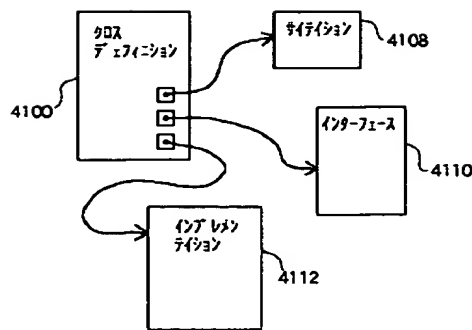
【図107】



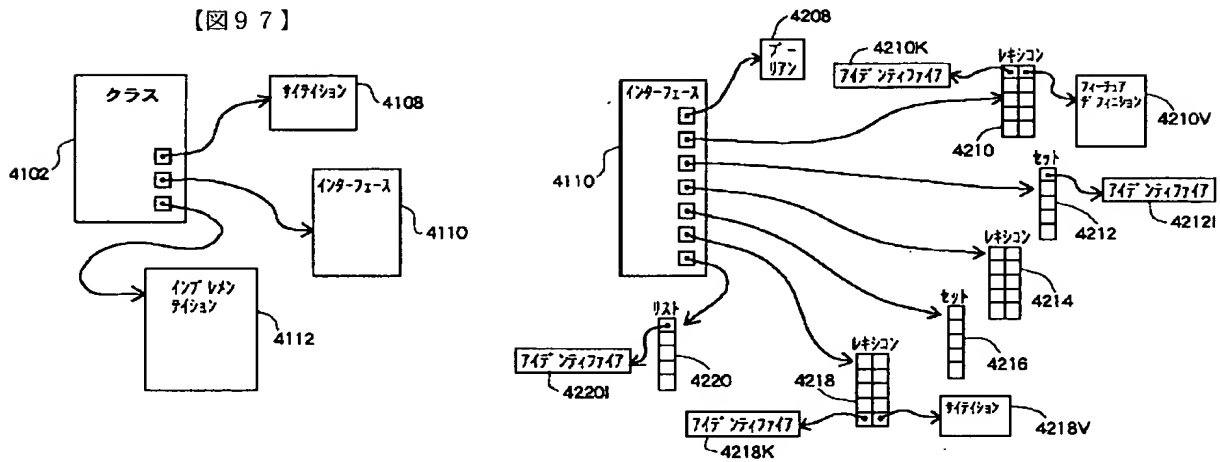
【図 9 4】



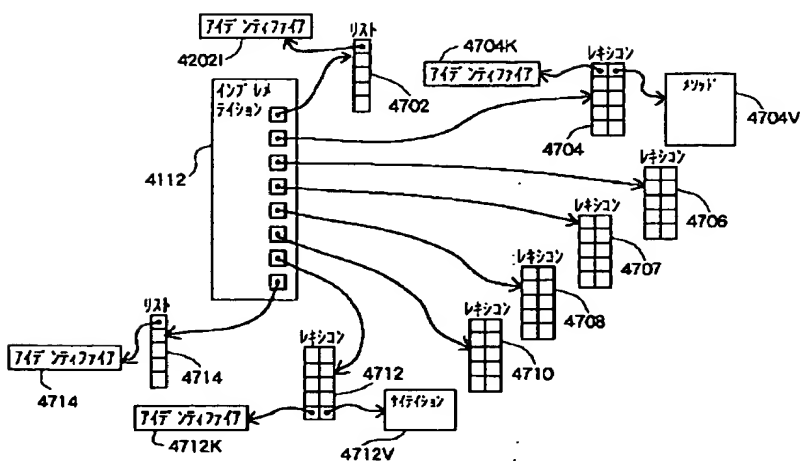
【图 9 6】



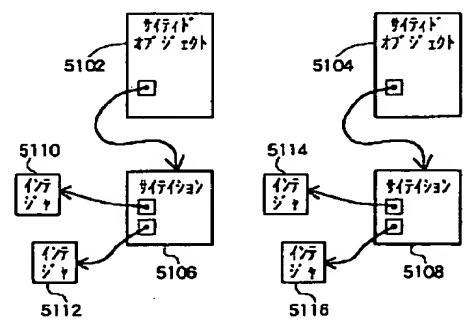
【図 9 9】



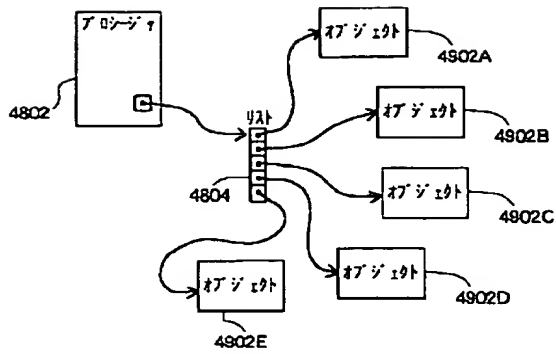
【図 104】



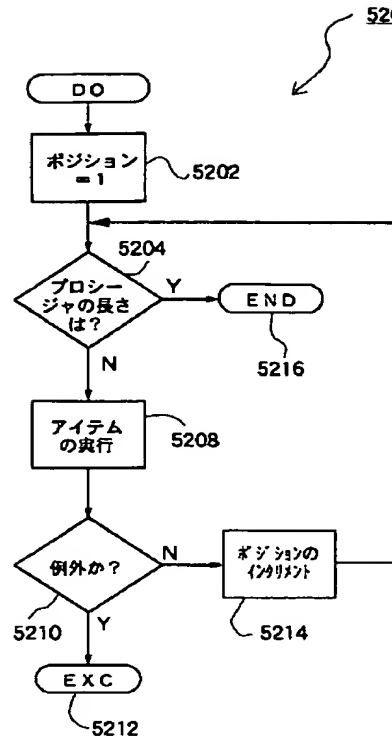
【図 108】



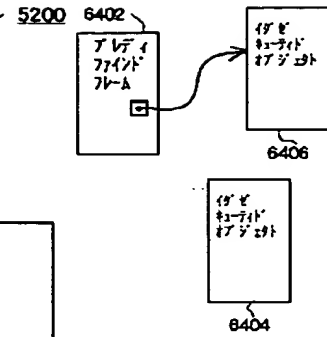
【図106】



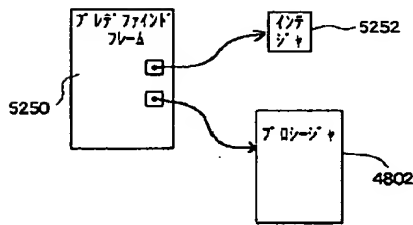
【図109】



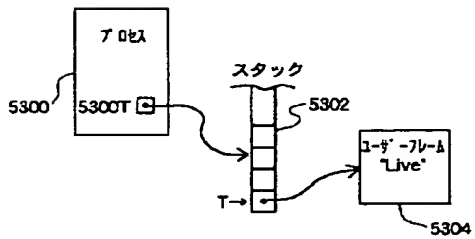
【図127】



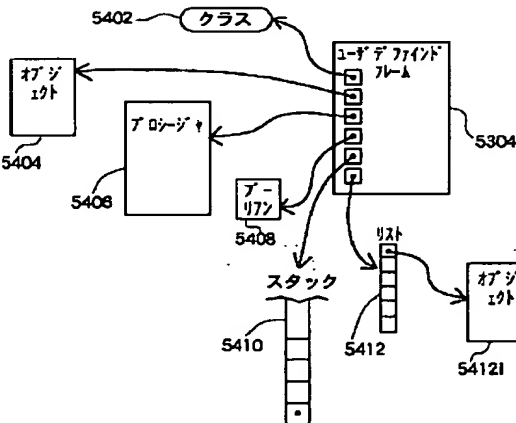
【図110】



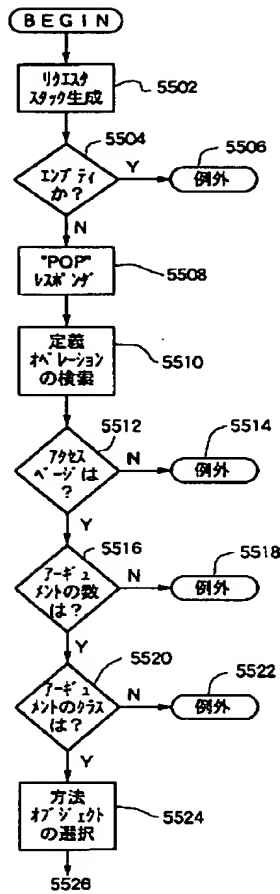
【図111】



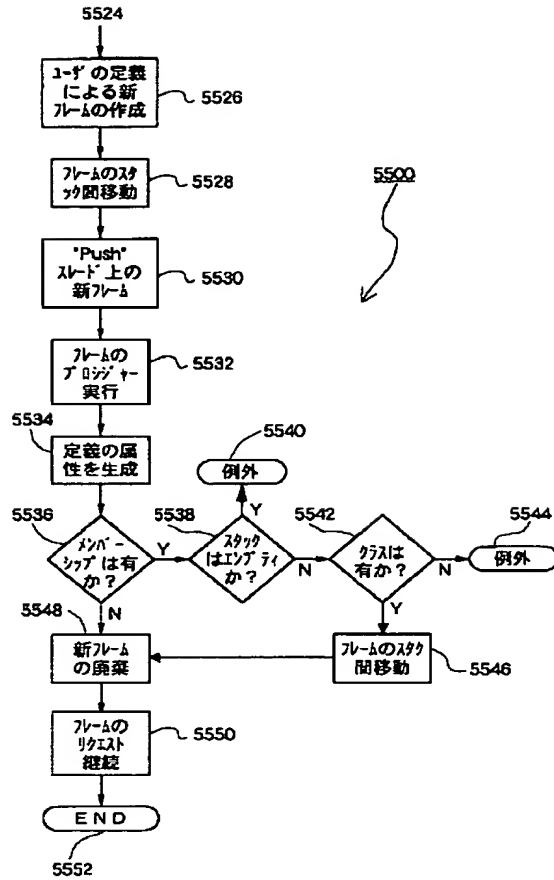
【図112】



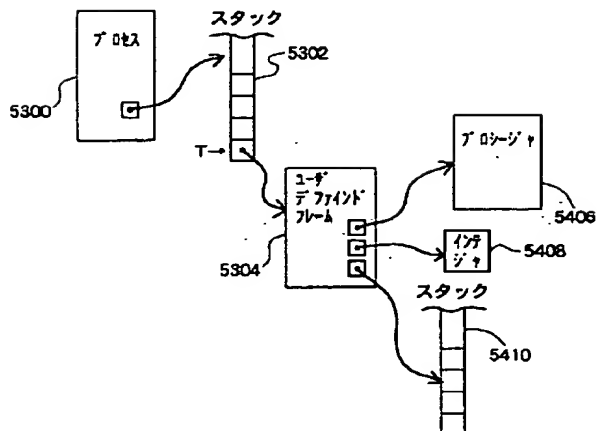
【図113】



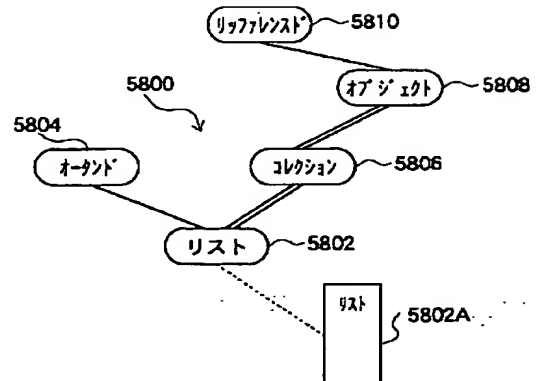
【図114】



【図115】

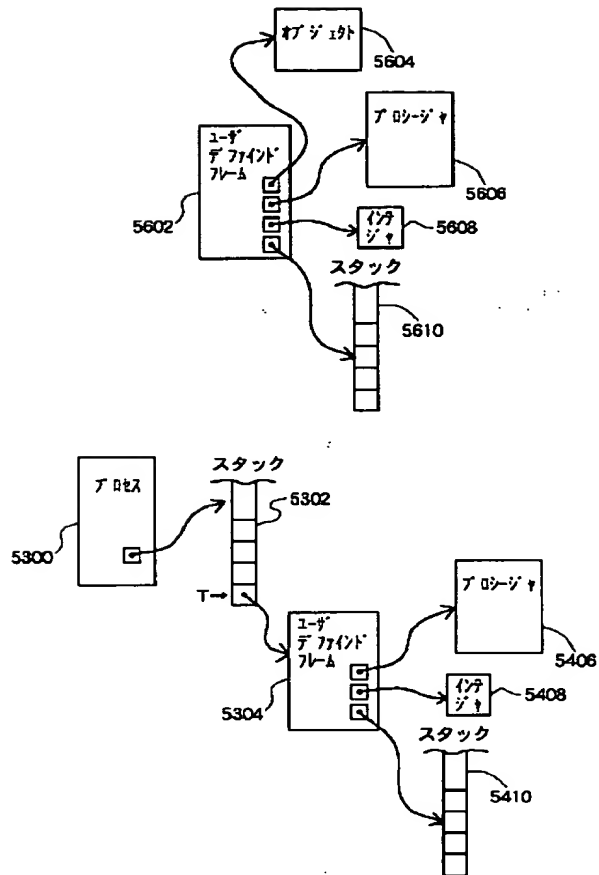


【図119】

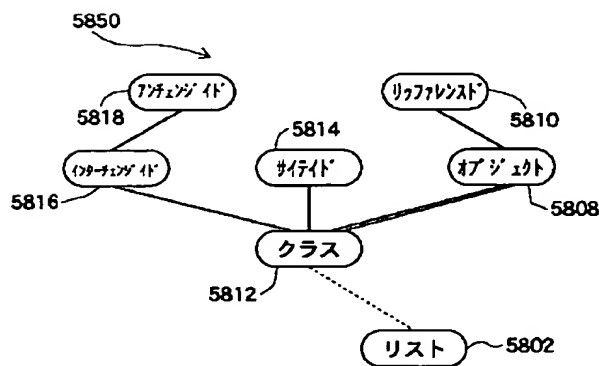




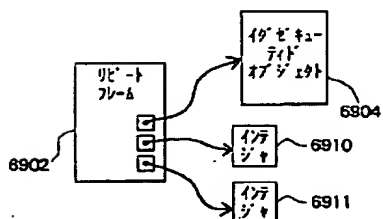
【図116】



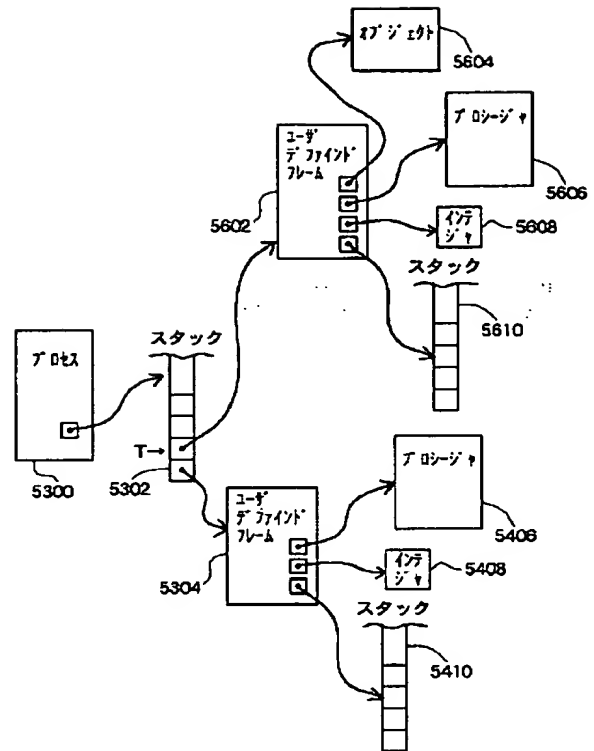
【図120】



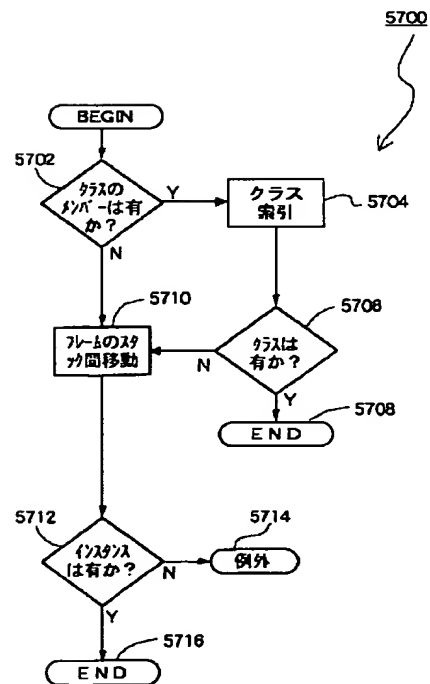
【図133】



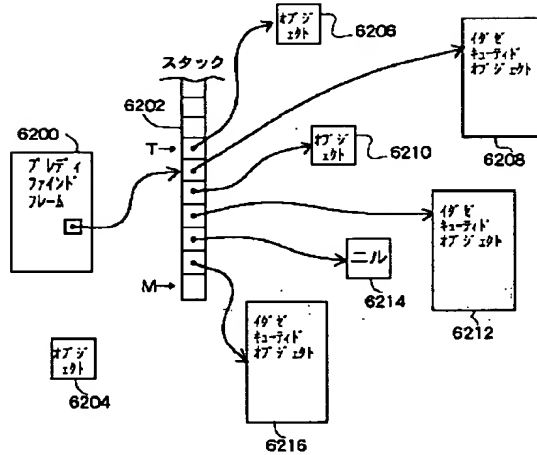
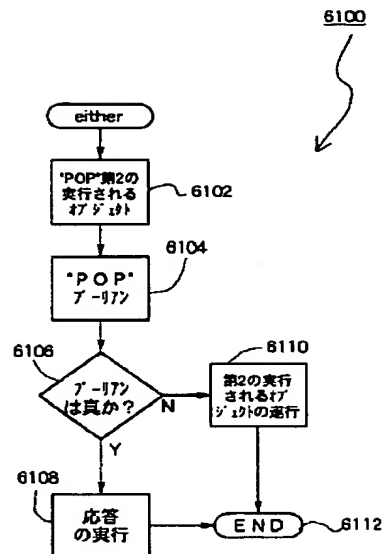
【図117】



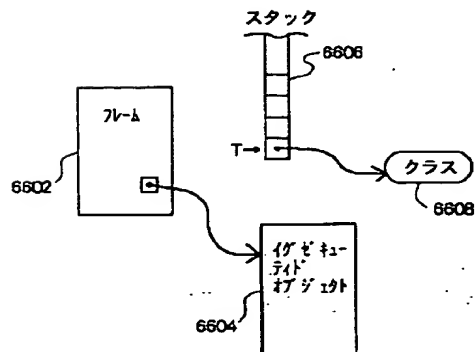
【図118】



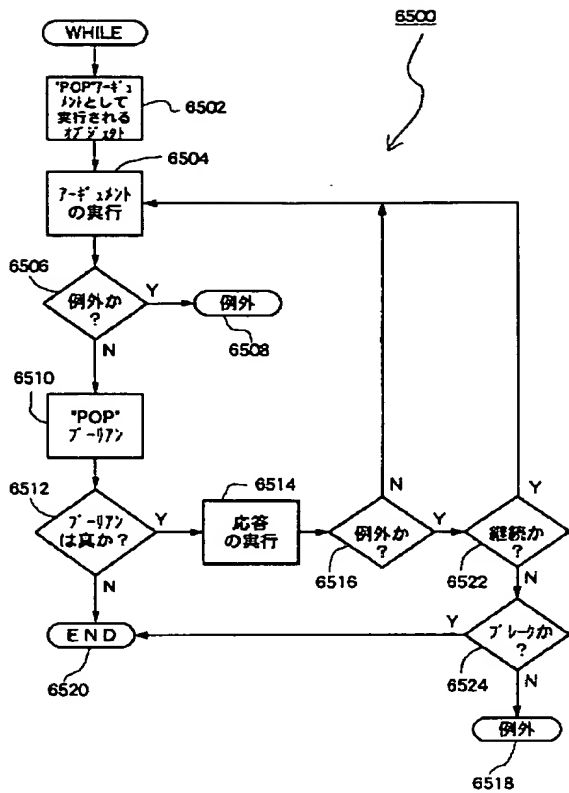
【图 1 2 3】



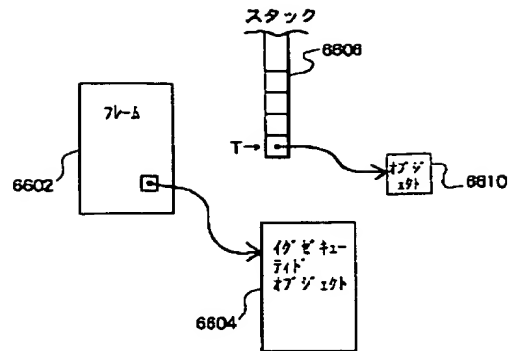
【図 129】



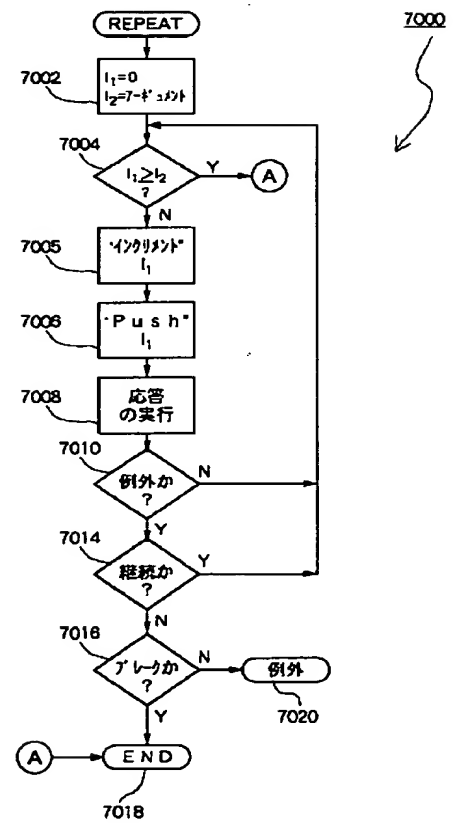
【図128】



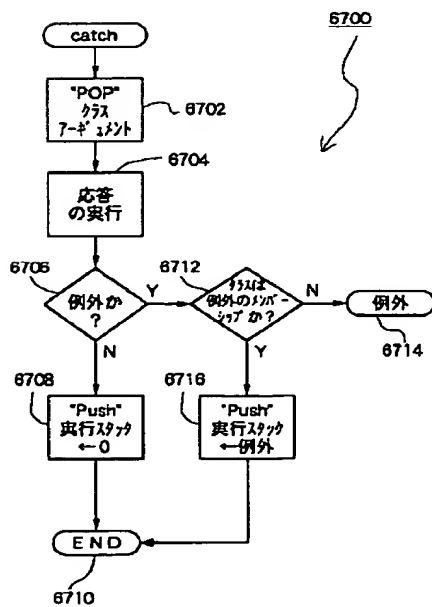
【図130】



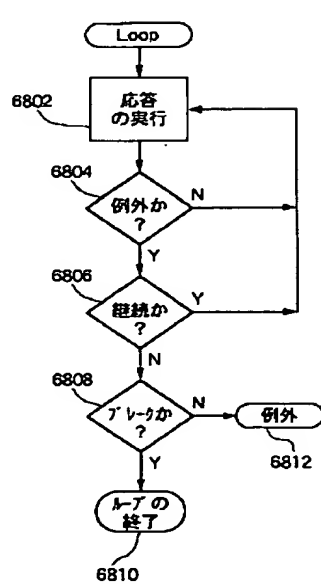
【図134】



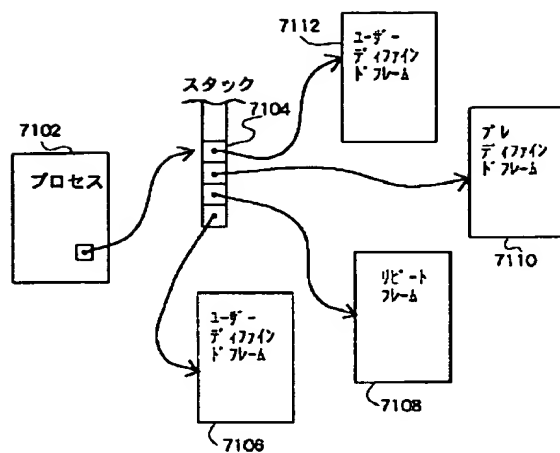
【図131】



【図132】



【図135】



フロントページの続き

(72) 発明者 クリストファー エス. ヘルゲッサン  
 アメリカ合衆国 カリフォルニア州  
 94070, マウンテン ビュー, パーシ  
 ティ コート 1025

(72) 発明者 ダグラス エイ. ステードマン  
 アメリカ合衆国 カリフォルニア州  
 94070, マウンテン ビュー, ラッサ  
 ム ストリート #83, 2250